Virtual Private Blockchains for GDPR: Cheap Private Blockchains out of Public Ones

Diego Pennino[1], Maurizio Pizzonia[2]

[1]*Università degli Studi della Tuscia, Dipartimento di Economia, Ingegneria, Società e Impresa, Via del paradiso 47, 01100 Viterbo, Italy;*

[2]*Università degli Studi Roma Tre, Dipartimento di Ingegneria, Sezione Informatica e Automazione, Via della Vasca Navale 79, 00146 Rome, Italy;*

Abstract

Compliance with privacy regulations, like the European GDPR, poses a big pressure toward the adoption of private blockchain with respect to solutions based on public blockchains. However, there are cases in which adopting a private blockchain is hardly viable for other reasons, leading to situations that might result in a big obstacle for blockchain adoption of whatever kind. In this paper, we describe an approach to make a *virtual private blockchain* on top of a regular public blockchain, obtaining the security of the consensus algorithm of the latter while keeping the privacy features of a private blockchain. Our approach leverages cryptographic zero-knowledge proofs and authenticated data structures to allow the underlying public blockchain to verify that transactions of the VPBC conform to its specific consensus rules, without decrypting the transaction itself.

1. Introduction

Blockchains are usually classified either as public or private. While public blockchains are famous for supporting the cryptocurrency ecosystems and to be a fundamental ingredient of the web3, private blockchains mostly find their applicative niche in restricted consortia or as the basis for integrity-critical application for company internal use.

There are advantages that are specific to private blockchains, like confidentiality, flexibility, efficiency, and easier control of costs. Maybe the most interesting one is the possibility to keep transactions confidential so that only the participants that contribute to the infrastructure of the private blockchain can see them. In fact, while many proposals are aimed to improve public blockchain with respect to efficiency, cost and developer friendliness, this is much harder for the confidentiality aspect. Confidentiality is required for compliance with privacy regulations (like the European GDPR [1]) together with the "right to be forgotten". Two requirements that are very hard to fulfill adopting a public blockchain. It is beyond doubt that, when personal data are involved, compliance with privacy regulations is a big pressure toward the choice of a private blockchain with respect to a public one. However, there are cases in which adopting a private blockchain is hardly viable for other reasons, leading to situations that might result in a big obstacle for blockchain adoption of whatever kind.

DLT24

diego.pennino@unitus.it (D. Pennino); pizzonia@ing.uniroma3.it (M. Pizzonia)
0000-0001-5339-4531 (D. Pennino); 0000-0001-8758-3437 (M. Pizzonia)

Consider a situation in which the subjects that are supposed to participate in a private blockchain are just in small numbers, e.g., three. The security of the whole blockchain is very poor, especially if two of them may have interest to collude. This is just a simplified model of many other possible situations in which

- nodes are hosted by a handful of big subjects (the only ones that can afford to do it), and that may have interest in controlling the blockchain system, and
- a large number of "small" subjects just participate as clients not being involved in consensus.

A notable case of this model is the one in which there is only one big player that can afford to host a node and a multitude of small players that may have interest to participate but cannot afford to be first class citizens hosting a node. In these situations, the adoption of a private blockchain is possible only if the trust relationships among subjects make the attack unlikely, and hence the main purpose for adopting a blockchain is to improve reliability.

Developing the above considerations, we can easily identify applicative contexts in which the adoption of blockchain, although desirable, is impossible. For example, it is nearly impossible to adopt a private blockchain to support cooperation among small and local companies, cooperation among private (non-company) subjects (e.g., for local self-organized committees), cooperation with short lifespan even among companies, tracking cooperation between one big subject and a multitude of small ones, e.g., companies managing delivery riders, like deliveroo [2], or companies providing visibility, tracking and other value added services to craft sectors, like Yhop [3].

The objective of this paper is to provide an architecture in which public blockchain can be used to realize a *virtual private blockchain* (VPBC) in which the security of the system is provided by the underlying public blockchain, and in particular the high number of subjects involved in consensus, while confidentiality is obtained by the adoption of suitable cryptographic primitives and zero-knowledge proofs. The final goal is to enable the use of public blockchain to support applications that have to be compliant with privacy regulations and have to be very cheap for all participants (or at least for the large majority of them but one).

The name VPBC is clearly inspired by the Virtual Private Network technology in which a private point-to-point connection is obtained by a clever use of cryptographic primitives on top of plain internet. This connection is cheap since involved parties do not have to install any infrastructure. In a similar way, we propose to create a private blockchain among a number of subjects from a public one without asking the subjects to keep costly nodes.

We provide an architecture that achieves most of the desired goals. We discuss how this can realized with currently available technology and cryptographic primitives. In particular, the main idea is inspired by the work in [4], and consists in moving all the checks and computations that are usually performed by the blockchain, on the client that is creating the transaction. The transaction of a VPBC is encrypted to keep confidentiality and equipped with a zero-knowledge succinct non-interactive proof that the encrypted content satisfies certain application-specific consensus rules. Clearly, the public blockchain has to check the correctness of the zero-knowledge proof to accept the transaction, which can be done by a smart contract. Note that this can be done without decrypting the content. In this way, transactions are kept private to the participants in the VPBC.

Other details are discussed in the paper, in particular regarding the synchronization of the shared state and the throughput and latency of this kind of approach.

The rest of the paper is organized as follows. In Section 2, we describe the background and the related works of our approach. In Section 3, we formalize the problem. In Section 4, we describe our architecture. In Section 5, we briefly discuss some properties of our architecture. In Section 6, we present our conclusions and possible future research directions.

2. State of the Art

As stated in the Introduction, although there is a clear division between public and private blockchains, not all use cases fit neatly into this division. Before examining works similar to our proposal, we describe what other techniques are close to our goal explaining why they are not suitable for it.

Without involving the scenario of blockchains, in literature, every time we need privacy and data integrity, we use Authenticated Data Structures (ADS) [5, 6]. However, a solution based exclusively on ADS demands that the application model be client-server, with an untrusted server. This type of solution can be challenging when structure updates need to be performed. Any new root hash must be securely broadcasted to all participants. The protocols for ensuring data propagation and integrity become too complex in an adversarial environment.

Moving away from the pure use of authenticated data structures leads us to choose among solutions that involve the use of a blockchain, to increase the security of the system without overcomplicating the protocols. In the blockchain, one of the most well-known techniques for performing interactions between parties, without all participants in the chain knowing all the details, is the state channel [7, 8], or its generalized version called Multi-Party Virtual State Channels [9, 10]. The main idea behind state channels is to achieve the same security guarantees as the blockchain, but at the same time minimize the number of required on-chain transactions. They present several challenges, but the main disadvantage is the need for all participants to be online and actively involved in the operation of the channel.

To overcome the issues concerning channel confidentiality, authenticity, finalization, and dispute resolution it was proposed Speedster [11]. The main idea of Speedster is that every user creates an off-chain account protected by the hardware-based enclave, an instance of a *Trusted Execution Environment* (TEE). The main drawback is that these types of solutions are targeted at transaction speed, not simplicity, and cost reduction.

Another approach that is natural to think about is the creation of a private blockchain over a VPN network. This approach however is not viable when the number of participants that can afford to host a node is very small.

Contributions with a similar purpose but different techniques are described in [12, 13]. In both works, the main disadvantage of the VPBC architecture is the additional overhead required for sending and validating VPBC transactions. The first work introduces the overhead by adding a consensus algorithm among all VPBC peers. The second work introduces the overhead by using the *secret sharing* technique, in which each transaction must be split into n different parts.

As mentioned in the Introduction, our proposed solution involves encrypting transactions to maintain confidentiality and utilizing a zero-knowledge succinct non-interactive proof. In recent years, there has been an increasing demand for data privacy, leading to the use of zero-knowledge proofs in the blockchain scenario. See [14, 15] for a survey. A zero-knowledge proof is a cryptographic protocol that allows the prover to prove the truth of a statement to the verifier without providing any additional information. Non-interactive zero-knowledge proofs are particularly useful when it is not possible for the two parties to interact. There are various types of non-interactive proof, with different efficiency tradeoffs. Among the currently best-known and most widely used approaches are: zk-SNARK [16, 17], zk-STARK [18], zk-SNORK [19, 20, 21], Bulletproofs [22].

Notably, zero-knowledge proofs are used in blockchain to implement ZK-rollup. A rollup [23] is a scaling solution that aggregates many transaction executions outside the main chain and sends the data back to the main network as a single transaction. By combining zero-knowledge with rollup protocols, we obtain so-called ZK-rollups, which allow us to publish a validity proof of the aggregate in the main chain without having to publish other information about individual transactions, thus reducing operation costs. A follow-up to the zk-rollup thread is the Zero-Knowledge Virtual Machine (ZKVM). Due to its high adoption, the Ethereum Virtual Machine is the one most studied for an evolution to a ZKEVM(zk-EVM) [24]. Since the classic Ethereum Virtual Machine does not support ZK proofs by default, zk-EVM aims to make it possible to create EVM-compatible zk-rollups.

A number of works adopt the above mentioned cryptographic primitives to achieve results that are somewhat akin to a VPBC and may be in principle used as a building block to build a VPBC. In [25], a privacy enabled cryptocurrency compatible with Ethereum is presented. The work in [26] proposes a primitive for decentralized private computations. The work in [27] proposes a privacy-preserving auditable blockchain for financial applications. The Aleo blockchain [28], is a blockchain specifically designed for privacy based on the zkVM approach. In addition, some similarities to a VPBC can be found in other work related to non-blockchain scenarios that use zero knowledge to ensure that some sort of policy is being followed correctly. In [4, 29], a Zero-knowledge Middleboxes network is presented that enforces network usage policies on encrypted traffic. Clients send the middlebox zero-knowledge proofs that their traffic is policy-compliant.

3. Problem Formalization

A set of m subjects $S = \{s_1, \ldots, s_m\}$ need to share privately a data structure A to support a certain application. The data structure A evolves over time when subjects emit transactions. We denote the i-th transaction as t_i. The state of A after t_i is denoted A_i. The application is defined by the *application rules* that state which transactions can be accepted. We do not go into the details of the application rules, but we suppose that, given certain application rules, there is an algorithm that given A_{i-1} and the candidate t_i says if t_i can be accepted, and, in such case, which is A_i. Subjects in S do not trust each other in the sense that each of them can have interest in subverting the application rules and/or tampering with A for its advantage. We also admit any collusion, even among $m - 1$ subjects against the remaining one.

Transactions and the data structure A have to be kept confidential.

For the sake of simplicity, we assume to have at disposal an ideal public blockchain which is secure, free, and can execute Turing complete smart contracts. Gas-like limitations are likely to hold in practice, however, this is not very relevant in this paper, hence we do not consider it in our model.

We look for a solution that leverages the public blockchain to realize a service that resembles a private blockchain among subjects of S preserving confidentiality.

4. Solution

In this section, we show an architecture that solves the problem described in Section 3. Table 1 summarizes the symbols used in this section.

We represent the data structure A with an *Authenticated Data Structure* (*ADS*) in the sense intended in [5, 6]. For simplicity, we denote the ADS associated with A with the same symbol A. We briefly recall the properties of an ADS. The content of A is uniquely associated with a cryptographic hash, called *root hash*. We denote a version of A with root hash r by A_r. It is possible to query A_r, whose root hash is r, and obtain the result data d with a proof p that d is in A_r. The proof p can be efficiently checked against r without knowing A_r. Suppose to update A_r applying some changes to d so that a new version $A_{r'}$ is obtained. Given r, p, and the changes to be applied to d, it is possible to calculate the root hash r' of $A_{r'}$ without knowing A_r. For simplicity we assume A is just a key-value mapping, hence data d is identified by just one or more keys.

Our architecture is shown in Figure 1. It is made of an *underlying* public blockchain U that we intend to exploit to support a VPBC V. The VPBC V is formed by a set of participants, which are untrusted to each other. For simplicity, we assume that they are all equivalent and that each of them stores A locally. Transactions in V are called *virtual transactions* or *vtx*'s and express changes to A. A vtx is encrypted so that it can be read only by participants of V and when it is accepted it is recorded encrypted in blocks of U. Participants of V update their local copy of A when a vtx is accepted by U. In the following, we explain the details of how it is possible for U to accept (or not) an encrypted vtx.

From the point of view of V, a vtx is a pair $v = \langle I, E \rangle$, where I is the set of the keys of A involved in v and of the value they have when v is supposed to be executed and E states the effect of the execution of v on a subset of I, that is the keys in I that are changed by the execution of v and the new value they assume. Not all vtx's can be accepted. Which ones can be accepted depends on the application. We suppose there is an algorithm $\mathcal{A}(v, A_r)$, associated with the application, that given a vtx v to be executed on the state A_r, first checks that I conforms to A_r (and fails otherwise) and then, looking solely to v, decides to accept v or to fail by performing all checks that depends on the application.

Adopting the above notation, the VPBC problems can be stated as follows: realize V on U exploiting the consensus of U so that only vtx's that conforms to $\mathcal{A}$ are accepted by U and making vtx's confidential by some form encryption.

To exploit the consensus mechanism of U while keeping vtx's confidential, we make use of zero-knowledge proofs [14]. In some sense, this approach is akin to the one described in [4] in which a device needs to take a decision on the content of an encrypted packet without knowing its content. There are many schemes of zero-knowledge proofs known in literature that are suitable for application in a blockchain context. In our description, we do not make any assumption besides the non-interactivity of a chosen scheme. A brief discussion on efficiency and applicability is given in Section 5.

We denote by $\{x\}$ the encrypted version of x by a symmetric key K shared among all participants of V and not published to anyone else. We assume each participant of V has associated a private key s, which can be used to sign a transaction or part of it, and a corresponding public key p. A certification authority is entitled to decide who are the participants of V and expresses this decision by signing the public key p of each participant with its secret key sk_c. The signed version is denoted $[p]_c$ and can be checked with the public key of the certification authority pk_c.

Given a vtx $v = \langle I, E \rangle$, we define the *augmented form* of v, denoted $\bar{v}$, to be the 5-tuple $\langle v, r, P(I), r', \sigma \rangle$ where

- r is the root hash of A_r on which v is supposed to act,
- $P(I)$ is a set of ADS proofs, one for each element of I, with respect to A_r,
- r' is the root hash of $A_{r'}$ resulting from the application of the effects E of v on A_r,
- σ is the signature of v by one of the participants of V.

Note that, it is easy to derive from $\mathcal{A}(v, A_r)$ an equivalent $\bar{\mathcal{A}}(\bar{v})$ that does not need to know A_r, by exploiting the proofs in $P(I)$ instead of accessing A_r.

We assume that the chosen zk scheme can be used to provide a *prover* function $\text{zkp}_{\bar{\mathcal{A}}}(\{\bar{v}\}, \bar{v}, K, [p]_c, pk_c)$, where $\bar{v} = \langle v, r, P(I), r', \sigma \rangle$, returning a non-interactive zero knowledge proof z that

- $\{\bar{v}\}$ is the correct encryption of $\bar{v}$ with key K,
- v is accepted by $\bar{\mathcal{A}}$ (and hence by $\mathcal{A}$),
- its execution transforms A_r into $A_{r'}$,
- σ is a correct signature of v, verified by public key p and $[p]_c$ is correctly verified by pk_c.

We also assume the chosen scheme provides a *verifier* function $\text{verify}(z, \{\bar{v}\}, r, r', pk_c)$ to check the soundness of z, its coherency with the root-hash r, and with the public key pk_c of the certification authority.

Note that, to accept v in V, we need to be sure that v can be accepted according to the application rules, signature is sound and that the state-related root hash r and r' are sound. Hence, v can be accepted if $\text{verify}(z, \{\bar{v}\}, r, r', pk_c)$ accepts z.

We are now ready to describe the functioning of the VPBC V over the underlying BC U. We assume that a smart contract C is available in U that has only one callable operation *execute*, whose signature is $\text{execute}(z, \{\bar{v}\}, r')$, it keeps in its state $\bar{r}$ which is the current root-hash of A, and knows pk_c. The *execute* operation performs verification using the above defined verification function called as follows: $\text{verify}(z, \{\bar{v}\}, \bar{r}, r', pk_c)$. If verification is successful, the operation call, comprising the $\{\bar{v}\}$ argument is recorded in a block of U and $\bar{r}$ is updated to match r'. If verification fails, no state change is performed and (possibly) nothing is recorded.

The algorithm $\mathcal{A}$, and its corresponding $\bar{\mathcal{A}}$, that express the application of V is unique and fixed for V. Also, the prover function $\text{zkp}_{\bar{\mathcal{A}}}(\cdots)$ and the corresponding $\text{verify}(\cdots)$ are fixed for V.

A participant to V, with private and public keys s and p, submits a vtx adopting the following procedure, which assumes that the root-hash r of the local copy of A corresponds to the root-hash $\bar{r}$ kept by C.

1. Prepare $v = \langle I, E\rangle$, where I is prepared on the basis of the local copy of $A = A_r$ and E records the changes to A to be applied by the vtx. Clearly, the changes have to be computed respecting the application rules, so that $\mathcal{A}$ can accept it.
2. Prepare $\bar{v} = \langle v, r, P(I), r', \sigma\rangle$, where ADS proofs in $P(I)$ are extracted from A_r, r' is computed by applying E to A_r (or to proofs in $P(I)$), and σ is the signature of v with the public key p of the participant.
3. Encrypt $\bar{v}$ by the symmetric key K obtaining $\{\bar{v}\}$.
4. Compute the zk proof $z = \text{zkp}_{\bar{\mathcal{A}}}(\{\bar{v}\}, \bar{v}, K, [p]_c, pk_c)$.
5. Submit the vtx by calling $\text{execute}(z, \{\bar{v}\}, r')$ on smart contract C.

If C accepts the vtx, all participants of V will get a block recording the call to $\text{execute}(\cdots)$ on smart contract C containing $\{\bar{v}\}$. We suppose that each participant to V has an instance of A with root-hash that is the one stored by C before accepting the vtx. The participant can decrypt $\{\bar{v}\}$ by K and obtain E, which can be used to obtain the new version of A that has root hash matching the one stored by C.

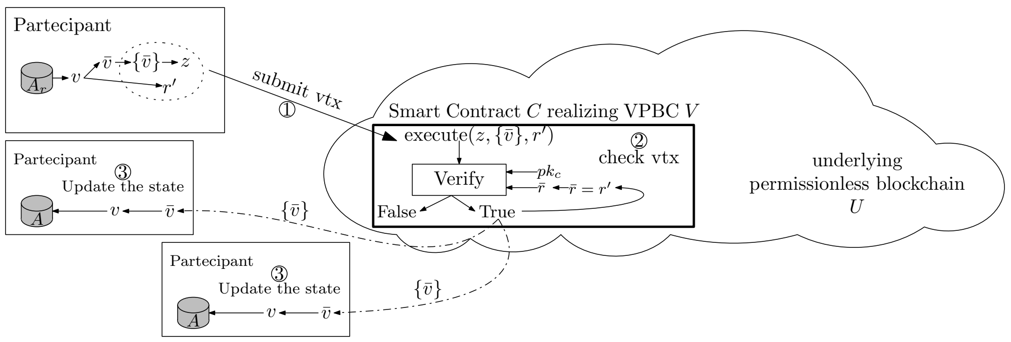

Figure 1: Virtual Private Blockchain Architecture.

5. Discussion

There are a number of aspects that can be improved, and/or studied, with respect to the very brief description provided in Section 4.

Cost. We assumed that all participants keep A and are always on-line to receive all accepted vtx's. However, for many of the scenarios described in the introduction, this is not convenient or too costly. Ideally, we would like the participants to be able to stay off-line and sync when on-line again. This is possible by (1) either gathering all missed vtx's or (2) asking to another participant to help to sync A to the last version. Both solutions have pros and cons. The first requires being able to identify the vtx's among all the transactions of the underlying blockchain. Ideally, this might be supported by the underlying blockchain or the smart contract C may keep track of them at least up to a certain extent. The second requires having at least one participant that is always on-line. This is compatible with scenarios in which there is at least one big player that has enough technical capability to do so.

Thruoghput. Currently, in our proposal, it is possible to perform only one vtx per block accepted in the underlying blockchain. While this might be acceptable in the scenarios mentioned in the introduction, it is clearly suboptimal. To improve throughput, a possible direction of study is to aggregate transactions with techniques that aggregate changes to an authenticated data structure, possibly referred to different, but recent, root-hashes. A similar approach was proposed in [30] in a different context. This might require to include the proofs $P(I)$ of a vtx unencrypted. Hence, an analysis of the possible confidentiality threats has to be performed.

Privacy Regulation Compliance. Our approach provides relevant steps toward a full compliance of private blockchain-based applications that takes advantage of premissionless infrastructures. First of all, the identification of *controller* and *processor* are streamlined as a VPBC greatly resembles a regular private/private blockchain. The only difference is that the infrastructure is provided by an external entity (the public blockchain) that while it does not guarantee confidentiality, it performs provably correct execution by design. The confidentiality aspect is compensated by the adoption of a proper form of encryption. Ensuring the principle of the right to be forgotten is slightly more complex. Suppose the controller is identified by the whole set of participants to the VPBC and suppose that a data subject has expressed its will for his/her personal data to be removed. The data A can be easily removed by a proper vtx[1] A bit more complex to address is the fact that vtx's containing personal data are permanently stored in the underlying blockchain, even though encrypted. To address this problem all participants have to forget the shared key K and agree on a new shared key K' to be used for the subsequent vtx's.

A possible Zero-Knowledge Protocol choice. Among the many ZK protocols described in literature, we think that it is worth considering the Groth16 [16] approach as a basis for a first prototype of our VPBC. Further, the availability of a framework like Circom [31] may greatly facilitate the creation of simple provers for the first experiments.

6. Conclusions and Future Work

We described an architecture to realize a virtual private blockchain, that is a blockchain in which transactions and state are confidential to a limited set of participants while the infrastructure is an underlying public blockchain. In particular, in our proposal, the security of the consensus is bound to the security of the consensus of the underlying blockchain. We identified a number of scenarios in which this would be desirable, i.e., when privacy regulation compliance is needed and the number of participants that can afford to set up a node is limited. Our architecture is based on encrypted (virtual) transactions and zero-knowledge proofs. We also discussed a number of aspects that our brief description does not handle which will be the subject of future research work.

[1]Any copy that a participant keeps outside any blockchain-related system is clearly outside the scope of this paper and the participant is legally responsible for it.

Symbol	Description
$\mathcal{A}$	Application algorithm
A	Shared Authenticated Data Structure
A_i	State of A after the application of transaction t_i
A_r	A with root hash r
C	the smart contract used to verify the correctness of $\bar{v}$
E	the new values of all keys in I
K	Symmetric key shared among all S
I	set of keys of A involved in v and of the value they have when v is supposed to be executed.
m	Cardinality of S
p	public key of a subject or ADS proof depending on the context
$P(I)$	is a set of ADS proofs, one for each element of I
pk_c	public key of the certification authority
r	root hash
S	Set of subjects participating in the VPBC.
s_i	i-th subject
s	private key of a subject
sk_c	private key of the certification authority
t_1	i-th Transaction
U	underlying public Blockchain
V	virtual private blockchain
v	virtual transaction
$\bar{v}$	augmented form of v, the 5-tuple $\langle v, r, P(I), r', \sigma \rangle$
z	non-interactive zero knowledge proof
$[p]_c$	signed version of p, signed with sk_c
$\{x\}$	the encrypted version of x by K
σ	the signature of v by one of the participants of V.

Table 1
Table of Symbols.

7. Acknowledgment

We are grateful to Pierpaolo Della Monica and Marco Zecchini for their valuable comments and discussions.

References

[1] E. Union, General data protection regulation (gdpr), regulation 2016/679, 2016.
[2] Wikipedia, Deliveroo ltd., https://en.wikipedia.org/wiki/Deliveroo, 2024. (Accessed on 11/03/2024).
[3] T. srl, Yhop, https://app.yhop.beer/, 2024. (Accessed on 11/03/2024).
[4] P. Grubbs, A. Arun, Y. Zhang, J. Bonneau, M. Walfish, {Zero-Knowledge} middleboxes, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 4255–4272.
[5] R. Tamassia, Authenticated data structures, in: Algorithms-ESA 2003: 11th Annual Euro-

pean Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11, Springer, 2003, pp. 2–5.

[6] A. Miller, M. Hicks, J. Katz, E. Shi, Authenticated data structures, generically, ACM SIGPLAN Notices 49 (2014) 411–423.

[7] J. Coleman, State channels - an explanation, https://www.jeffcoleman.ca/state-channels/, 2015. (Accessed on 05/03/2024).

[8] L. D. Negka, G. P. Spathoulas, Blockchain state channels: A state of the art, IEEE Access 9 (2021) 160277–160298. doi:10.1109/ACCESS.2021.3131419.

[9] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, K. Hostáková, Multi-party virtual state channels, in: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38, Springer, 2019, pp. 625–656.

[10] S. Dziembowski, S. Faust, K. Hostáková, General state channel networks, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 949–966.

[11] J. Liao, F. Zhang, W. Sun, W. Shi, Speedster: An efficient multi-party state channel via enclaves, in: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, 2022, pp. 637–651.

[12] A. Arena, P. Perazzo, G. Dini, Virtual private ledgers: Embedding private distributed ledgers over a public blockchain by cryptography, in: Proceedings of the 23rd International Database Applications & Engineering Symposium, 2019, pp. 1–9.

[13] S. Onalo, D. Gc, E. Pfluegel, Virtual private blockchains: security overlays for permissioned blockchains, Fifth International Conference on Cyber-Technologies and Cyber-Systems: CYBER 2020 (2020).

[14] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, X. Peng, A survey on zero-knowledge proof in blockchain, IEEE network 35 (2021) 198–205.

[15] J. Partala, T. H. Nguyen, S. Pirttikangas, Non-interactive zero-knowledge for blockchain: A survey, IEEE Access 8 (2020) 227945–227961.

[16] J. Groth, On the size of pairing-based non-interactive arguments, in: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35, Springer, 2016, pp. 305–326.

[17] M. Petkus, Why and how zk-snark works, arXiv preprint arXiv:1906.07221 (2019).

[18] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, Scalable, transparent, and post-quantum secure computational integrity, Cryptology ePrint Archive (2018).

[19] M. Maller, S. Bowe, M. Kohlweiss, S. Meiklejohn, Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 2111–2128.

[20] A. Gabizon, Z. J. Williamson, O. Ciobotaru, Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge, Cryptology ePrint Archive (2019).

[21] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, D. Song, Libra: Succinct zero-knowledge proofs with optimal prover computation, in: Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22,

2019, Proceedings, Part III 39, Springer, 2019, pp. 733–764.
[22] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in: 2018 IEEE symposium on security and privacy (SP), IEEE, 2018, pp. 315–334.
[23] L. T. Thibault, T. Sarry, A. S. Hafid, Blockchain scaling using rollups: A comprehensive survey, IEEE Access 10 (2022) 93039–93054. doi:10.1109/ACCESS.2022.3200051.
[24] Alchemy, What is a zkevm?, https://www.alchemy.com/overviews/zkevm, 2022. (Accessed on 07/03/2024).
[25] B. Bünz, S. Agrawal, M. Zamani, D. Boneh, Zether: Towards privacy in a smart contract world, in: International Conference on Financial Cryptography and Data Security, Springer, 2020, pp. 423–443.
[26] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, H. Wu, Zexe: Enabling decentralized private computation, in: 2020 IEEE Symposium on Security and Privacy (SP), IEEE, 2020, pp. 947–964.
[27] N. Narula, W. Vasquez, M. Virza, {zkLedger}:{Privacy-Preserving} auditing for distributed ledgers, in: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), 2018, pp. 65–80.
[28] A. N. FOUNDATION, Aleo | fully private applications, https://aleo.org/, 2024. (Accessed on 27/02/2024).
[29] C. Zhang, Z. DeStefano, A. Arun, J. Bonneau, P. Grubbs, M. Walfish, Zombie: Middleboxes that don't snoop, Cryptology ePrint Archive (2023).
[30] M. Bernardini, D. Pennino, M. Pizzonia, Blockchains meet distributed hash tables: Decoupling validation from state storage, CEUR Workshop Proceedings 2334 (2019) 43 – 55. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85063259759&partnerID=40&md5=484bac13f164c622fe00c8541b0ec436.
[31] iden3, Circom, https://iden3.io/circom, 2024. (Accessed on 12/03/2024).