Research Article

A survey on zero knowledge range proofs and applications

Eduardo Morais[1] · Tommy Koens[1] · Cees van Wijk[1] · Aleksei Koren[1]

Abstract
In last years, there has been an increasing effort to leverage distributed ledger technology (DLT), including blockchain. One of the main topics of interest, given its importance, is the research and development of privacy mechanisms, as for example is the case of zero knowledge proofs (ZKP). ZKP is a cryptographic technique that can be used to hide information that is put into the ledger, while still allowing to perform validation of this data. In this work we describe different strategies to construct zero knowledge range proofs (ZKRP), as for example the scheme proposed by Boudot (in: Bart (ed) Advances in cryptology—EUROCRYPT 2000, Springer, Berlin, 2000) in 2001; the one proposed by Camenisch et al. (in: Josef (ed) Advances in cryptology—ASIACRYPT 2008, Springer, Berlin, 2008), and bulletproofs (Bünz et al., in: 2018 IEEE symposium on security and privacy (SP), 2018), proposed in 2017. We also compare these strategies and discuss possible use cases. Since bulletproofs (Bünz et al. 2018) is the most efficient construction, we will give a detailed description of its algorithms and optimizations. Bulletproofs is not only more efficient than previous schemes, but also avoids the trusted setup, which is a requirement that is not desirable in the context of DLT and blockchain. In case of cryptocurrencies, if the setup phase is compromised, it would be possible to generate money out of thin air. Interestingly, bulletproofs can also be used to construct *generic* ZKP, in the sense that it can be used to prove generic statements, and thus it is not only restricted to ZKRP, but it can be used for any kind of proof of knowledge. Hence Bulletproofs leads to a more powerful tool to provide privacy for DLT. Here we describe in detail the algorithms involved in Bulletproofs protocol for ZKRP. Also, we present our implementation, which was open sourced (Morais et al., in: Zero knowledge range proof implementation, 2018. https://github.com/ing-bank/zkrangeproof).

Keywords Zero knowledge proof · Range proof · Set membership · Privacy · Blockchain

1 Introduction

DLT and blockchain have been subject to intense research in last years, because it allows to construct consensus among parties that do not fully trust each other, without the necessity of a trusted third party. However, in public and permissionless ledgers, transactions can be viewed by everyone in the network. This fact is a hindrance that we must overcome if those transactions contain privacy-sensitive information.

In order to protect private information, a possible alternative is to use a Trusted Execution Environment (TEE), like Intel SGX [39] technology. The idea is that any private data must appear in the blockchain in encrypted form. Only the owners of the subjacent cryptographic keys will be able to decrypt it. Validation of this information must be done in the TEE system, where the cryptographic keys can be embedded. Therefore, private data will only be visible after decryption, which occurs inside a controlled environment. Putting differently, a TEE offers protection against information leakage by restricting manipulation of private data to a region of memory that can not be accessed by other processes in the same machine, or even by its administrator. Nevertheless, attacks [21, 58] to SGX where proposed in literature, showing that this technology is vulnerable to *branch prediction* and *side-channel* attacks, respectively.

✉ Eduardo Morais, eduardo.morais@gmail.com | [1]ING, Amsterdam, The Netherlands.

Received: 28 December 2018 / Accepted: 25 July 2019 / Published online: 31 July 2019

A different approach to secure private data is ZKP, which is a cryptographic technique that have been used to provide *privacy by design* in the context of DLT and blockchain. Shortly, ZKP allows an entity called *prover* to argue to another party, called *verifier*, that a determined statement is true without revealing more information than strictly necessary to convince her.

In previous works [40, 48] ING described some preliminary results. The purpose of this work is to extend them in order to provide a complete survey on ZKRP protocols.

In summary, ZKRP allows to prove that a secret integer belongs to a certain interval. For example, if we define this interval to be all integers between 18 and 200, a person can use the ZKRP scheme to prove that she is over 18. This gives her permission, according to some regulation, to consume a determined service, but without revealing her specific age. In the context of payment systems, if party *A* wants to transfer money to party *B*, then it is possible to utilize ZKRP to prove that the amount of money in the transaction is positive, otherwise, if the amount is negative, such transaction would in fact transfer money in the opposite direction, i.e. from *B* to *A*.

In the following sections we describe in detail the algorithms necessary to implement ZKRP and instantiate the underlying parameters in order to obtain an appropriate level of security. We also compare the different schemes with regards to proof size, and the complexity of the prover and verifier algorithms.

1.1 Contributions

There are many surveys about zero knowledge proofs, but mostly related to the theoretical foundations of the proposed cryptographic constructions. The main goal of this survey is to bridge the gap between those papers whose audience is the cryptographic community and the community of developers that are more focused on implementation aspects. There is currently an effort to standardize zero knowledge proofs [14], where academia and industry started the effort to produce a standard to implement ZKPs. In particular, some *ZK gadgets* were identified as important building blocks for the construction of solutions to more complex problems. Among the ZK gadgets that were discussed, we can remark ZK Range Proofs, ZK Set Membership and cryptographic accumulators. Although those primitives have some relation between each other, the focus of this work is on ZK Range Proofs. In Sect. 7 we discuss these related works and give references to the interested reader.

Next we summarize the main contributions of this work:

- Survey possible use cases for ZKRP and indicate which papers in the literature present important contributions for the construction of efficient solutions to this use cases.
- Describe in detail the algorithms required for representative constructions of ZKRP. In particular, we describe how the Fiat–Shamir must be implemented in order to obtain non-interactive protocols.
- Update the survey presented in the work by Canard et al. [13] with newer proposals.
- Present our open source implementation [49] of the algorithms detailed in Sect. 4.

1.2 Organization

In Sect. 2 we describe possible use cases for ZKRPs. In Sect. 3 we give fundamental results that are important to understand the rest of the document. In Sect. 4 we describe in detail how to implement ZKRP using different strategies. In Sect. 5 we describe our implementation, while in Sect. 6 we compare the schemes with respect to proof size, prover and verifier complexities. In Sect. 7 we discuss related work and give some final remarks.

2 Applications

In order to give the reader a motivation to investigate further on Zero Knowledge Proofs, we present in this section some interesting applications.

- *Over 18* ZKRP can be used to prove someone is over 18 without revealing her exact age. Thus it is possible to allow the person to consume some service without requiring her to show paper documents, which contain more information than necessary for the age validation. In this situation it is important to have a trusted party to generate a commitment, as described in subsection 3.2, which attests that the information contained in it is correct. The person can not generate the commitment by herself because a malicious user could utilize fake data to prove the desired statement, even though the real data does not respect that property.
- *Know Your Customer (KYC)* As explained above, ZKRP allows to validate that a determined piece of private information belongs to a numeric interval. This property may be used to ensure compliance, while preserving a client's privacy. For example, an interesting use case is the so-called *anonymous credentials*, where a trusted party can attest that a user credential contains attributes whose values are correct, allowing to prove certain properties in zero knowledge way.
- *Mortgage risk assessment.* It is possible to prove that the salary of an individual is above some threshold in order to get a mortgage approved. In general, threshold vali-

dation is a key verification that must be performed in financial risk assessment. Therefore ZKRP turns out to be very important for financial institutions.

- *Rating and investment grading* The problem of rating companies according to their level of productivity or financial health can be modeled by determining a partition of a numeric interval, given by a sequence of increasing numbers $A_0, A_1, \dots, A_k$, such that the highest score is attributed to companies rated above A_k (or sometimes below A_0). The company's health is measured to obtain a value x, and the resulting grade depends upon which sub-interval x belongs to. Therefore, it is necessary to verify if $x \in [A_i, A_{i+1})$ for each $0 \leq i \leq k$. As far as we know there is no research applying ZKRP to this specific problem, where maybe more efficient constructions could exist, when compared to the straightforward solution of using ZKRP $k + 1$times.
- *Electronic voting* This is an important topic of research, which attracted the attention of many researchers in last years. Different solutions [1, 26, 28, 36] were proposed to different types of elections. Some solutions are based on zero knowledge proofs, like ZKRP, proof of shuffling, proof of decryption and other related techniques, while others use different cryptographic primitives, like homomorphic threshold encryption and Multi-Party Computation (MPC).
- *Electronic auctions and procurement* Secure electronic auctions is a subject that has being focus of research for a long time [44], and it is an important motivation in the study of ZKRPs, since it is one of the main cryptographic techniques that can be used to construct secure protocols. In particular, it is possible to remark the proposal of secure constructions [47, 50, 54] for Vickrey auctions, where the winner pays the second highest bid. A complementary problem to electronic auctions is *procurement*, where parties concur for the lowest price. According to the World Bank report [59], the volume of bribes in public sector procurement is roughly US$200 billion per year.

The applications described above are general purpose, but could be interesting also in the context of DLT and blockchain technology. Next we focus on application that are important in the specific scenario of DLT and blockchain:

– *Confidential Transactions and Mimblewimble* In 2016, Confidential Transactions (CT) were proposed by Maxwell [46], which utilizes Pedersen commitments [51] to hide transactions amounts. Instead of publishing the amounts being spent in the clear, each party uses the commitment scheme to hide the amount, what makes it infeasible for an adversary to obtain any information about transaction denominations. Since a Pedersen commitment is homomorphic, it allows transactions outputs to be added up without requiring to open the subjacent commitments. Also, the commitment can be used to generate a ZKRP, which is sufficient to validate that a transaction is correct. For instance, it is necessary to show that the amount lies in the interval $[0, 2^n)$, where 2^n is considerably smaller than the size of the underlying group used to construct the Pedersen commitment, ensuring there is no overflow; and 2^n is big enough to deal with every possible valid denomination.

  However, the usage of ZKRP would make the size of transactions too big. Namely, CT with just two outputs and 32 bits of precision would require roughly a ZKRP whose size is 5 KB, leading to transactions whose total size is equal to 5.4 KB@. Thus, ZKRP would correspond to almost 93% of the transaction size. Therefore in order to use CT in Bitcoin, we would need 160 GB only for ZKRP@. If Bulletproofs where used in replacement of the underlying range proof used in CT, then it would reduce this requirement to only 17 GB.

  Mimblewimble [52] is an optimization to CT that can make the size of the ledger even smaller, by aggregating and compressing transactions in such a way that avoids the necessity to download old and unspent transactions outputs.

– *Provisions* Provisions [23] is a protocol that allows a Bitcoin exchange to prove it is solvent, by showing that each account has positive balance, and also showing that the exchange has an amount of funds that is larger than or equal to the summation of all individual account's balance in the system. The challenge here is to calculate a single zero knowledge proof based on the information provided by different participants. This is difficult because each individual balance is encrypted using distinct keys, thus combining them is not straightforward, and requires MPC. Bulletproofs has a MPC protocol that solves this problem efficiently. For instance, if we consider a cryptocurrency exchange with 2 million clients, current implementation of Provisions requires 62 MB of ZKRPs. However, using Bulletproofs this number can be reduced to less than 2 KB, which corresponds to an optimization factor of 300.

3 Fundamentals

In this section we define commitment schemes, zero knowledge proofs and other important components that are necessary in order to comprehend this work. The purpose of this section is not to present very formal definitions. To achieve this goal, the reader can use Goldreich's book [33].

Notation Notation $x \in_R S$ is used when variable x is set to a random element of set S. We are going to use Camenisch and Stadler [10] notation for proofs of knowledge:

$$\mathrm{PK}\{(\delta, \gamma) : y = g^{\delta}h^{\gamma} \wedge (u \leq \delta \leq v)\},$$

which denotes a proof of knowledge of integers δ and γ such that $y = g^{\delta}h^{\gamma}$ and $u \leq \delta \leq v$. In other words, this notation means that y is the commitment to the secret value δ, which is contained in the interval $[u, v)$. Greek letters are used to denote values that must be known only to the prover. For instance, we have that δ is her private data, while γ is a random value that is used to hide δ.

Finally, we use notation $x \overset{?}{=} y$ to check if x is equal or not to y.

3.1 Assumptions

The constructions presented in this paper are based on the assumptions described in this section.

The strong RSA assumption first appeared in the work of Fujisaki and Okamoto [31]. It is a stronger assumption with respect to the conventional RSA assumption, because any adversary who can break the RSA assumption would also be able to break the strong RSA assumption. In [22] it is shown how to replace the strong RSA assumption by the standard RSA assumption in many ZKP application including ZKRP.

Definition 1 (*RSA assumption*) Given RSA modulus n, RSA exponent e and an element $y \in \mathbb{Z}_n^{\star}$, it is infeasible to find integers x such that $y = x^e \pmod{n}$.

Definition 2 (*Strong RSA assumption*) Given an RSA modulus n and an element $y \in \mathbb{Z}_n^{\star}$, it is infeasible to find integers $e \neq \pm 1$ and x, such that $y = x^e \pmod{n}$.

Definition 3 (*Discrete Logarithm assumption*) Let $\mathbb{G}$ be a group of prime order q, a generator $g \in \mathbb{G}$ and an arbitrary element $y \in \mathbb{G}$, it is infeasible to find $x \in \mathbb{Z}_q$, such that $y = g^x$.

Definition 4 (*q -Strong Diffie-Hellman assumption*) Given groups $\mathbb{G}_1$ and $\mathbb{G}_T$, associated with a secure bilinear pairing map e; given generator $g \in \mathbb{G}_1$ and powers $g^x, \ldots, g^{x^q}$, for $x \in_r \mathbb{Z}_p$, we have that it is infeasible for an adversary to output $(c, g^{1/(x+c)})$, where $c \in \mathbb{Z}_p$.

It is important to remark that these assumptions are not valid if quantum computers come to existence. Therefore, the research of quantum-resistant ZKPs is a very important subject.

3.2 Commitment

Shortly, a cryptographic commitment allows someone to compute a value that hides some message without ambiguity, in the sense that no one later will be able to argue that this value corresponds to a different message. In other words, given the impossibility to change the hidden message, we say that the user committed to that message. The purpose of using a commitment scheme is to allow a prover to compute zero knowledge proofs where the hidden message is the underlying witness w.

Definition 5 A **commitment scheme** is defined by algorithms Commit and Open as follows:

- $c = \mathrm{Commit}(m, r)$. Given a message m and randomness r, compute as output a value c that, informally, hides message m and such that it is hard to compute message m' and randomness r' that satisfies $\mathrm{Commit}(m', r') = \mathrm{Commit}(m, r)$. In particular, it is hard to invert function Commit to find m or r.
- $b = \mathrm{Open}(c, m, r)$. Given a commitment c, a message m and randomness r, the algorithm returns true if and only if $c = \mathrm{Commit}(m, r)$.

A commitment scheme has 2 properties:

- *Binding* Given a commitment c, it is hard to compute a different pair of message and randomness whose commitment is c. This property guarantees that there is no ambiguity in the commitment scheme, and thus after c is published it is hard to open it to a different value.
- *Hiding* It is hard to compute any information about m given c.

A well known commitment scheme is called *Pedersen commitment* [51]. Given group $\mathbb{Z}_p$, of prime order p, where the discrete logarithm problem is infeasible, the commitment is computed as follows:

$$c = \mathrm{Commit}(m, r) = g^m h^r.$$

In order to open this commitment, given message m and randomness r, we simply recompute it and compare with c. An interesting property is that Pedersen commitment is *homomorphic*. Namely, we have that for arbitrary messages m_1 and m_2 and randomness r_1 and r_2, such that $c_i = \mathrm{Commit}(m_i, r_i)$ for $i \in \{1, 2\}$, then

$$c_1 \cdot c_2 = \mathrm{Commit}(m_1 + m_2, r_1 + r_2).$$

Pedersen commitment is commonly implemented using groups over elliptic curves. Also, it is important to remark that if the discrete logarithm of h with respect to g is known, then it is easy to generate m' and r' such that

$\text{Commit}(m', r') = \text{Commit}(m, r)$, breaking the binding property. Thus in order to generate h securely, we must use a hash function that maps binary public strings to elliptic curve points [6].

Another commitment scheme that will be required later in this document is the Fujisaki-Okamoto commitment [31]. The formula to calculate the commitment itself is the same as in Pedersen commitment, namely $g^m h^r$. The difference is the underlying group, which for the Fujisaki-Okamoto is given by an RSA group $\mathbb{Z}_n$, where $n = pq$ and p and q are *safe primes*, what means that $(p-1)/2$ and $(q-1)/2$ are also prime numbers. Also, we have that the domain over which randomness r is chosen is different, because the Fujisaki-Okamoto commitment requires $r \in [2^{-s}n + 1, 2^s n - 1]$, with s chosen in such a manner that 2^{-s} is negligible. Interestingly, in the original paper [31] Fujisaki and Okamoto propose an interactive protocol for Zero Knowledge Range Proofs, but unfortunately the performance is not good for practical usage.

3.3 Zero knowledge proofs

Zero knowledge proofs (ZKP) were proposed in 1989 by Goldwasser, Micali and Rackoff [34]. Using this kind of cryptographic primitive it is possible to show that some statement is true about a secret data, without revealing any other information about the secret beyond this statement. Since then, ZKP became an important field of research, because it provides a new characterization of the complexity class NP, using the so-called *interactive programs*, and also because it is very useful to construct many cryptographic primitives. Given an element x of a language $\mathcal{L} \in NP$, an entity called *prover* is able to convince a verifier that x indeed belongs to $\mathcal{L}$, i.e. there exists a witness w for x. In particular we are interested in *proof of knowledge* (PoK), where the prover not only convinces about the existence of some witness, but also shows that the prover in fact knows a specific witness w. A desirable characteristic of such proof systems is *succinctness*, informally meaning that the proof size is small and thus can be verified efficiently. Such constructions are called zk-SNARKs [37]. However, although asymptotically good, zk-SNARKs still have some limitations and for specific problems it turns out that different approaches achieve better performance.

Nowadays ZKP is being used to provide privacy to DLT and blockchain. For instance, it allows to design private payment systems. In summary, we would like to permit parties to transfer digital money, while hiding not only their identities but also the amount being transferred, known as *denomination*. ZKP can be used to hide this information, but still permitting validation of transactions. An important validation is showing that the denomination is positive, otherwise some payer would be able to receive money by using negative amounts. In this context we have that zk-SNARKs don't provide good performance when compared to protocols designed specifically for this purpose. The focus of this document is the description of different constructions of ZKRP and compare them to understand when to use each scheme in practice. More concretely, ZKRP allows some party Alice, known as the *prover*, and who possesses a secret δ, to prove to another party Bob, known as the *verifier*, that δ belongs to the interval $[u, v)$, for arbitrary integers u and v.

Definition 6 A **Non-Interactive Zero Knowledge (NIZK) proof** scheme is defined by algorithms Setup, Prove and Verify as follows:

- Setup algorithm is responsible for the generation of parameters. Concretely, we have that $\text{params} = \text{Setup}(\lambda)$, where the input is the security parameter λ and the output is the parameters of the ZKP system of algorithms.
- Prove syntax is given by $\text{proof} = \text{Prove}(x, w)$. The algorithm receives as input an instance x of some NP-language $\mathcal{L}$, and the witness w, and outputs the zero knowledge proof.
- Verify algorithm receives the proof as input and outputs a bit b, which is equal to 1 if the verifier accepts the proof.

It is important to remark that not all ZKP schemes are non-interactive. On contrary, most ZKP protocols described in the literature are in fact interactive. In general, the prover must answer *challenge messages* sent by the verifier in order to convince him that the proof is valid, what requires multiple rounds of communication. In the context of DLT and blockchain applications, we would like to avoid this communication, because either (i) validating nodes can not properly agree on how to choose those challenges, since in many constructions we have to choose them randomly, while the verification algorithm must be deterministic in order to reach consensus; or (ii) because it would make the communication complexity of the system very poor. Nevertheless, the Fiat–Shamir heuristic [30] is a generic technique that allows to convert interactive ZKP schemes into non-interactive protocols. The drawback of this heuristic is that it makes the cryptosystem secure under the *random oracle model* [4] (ROM). In particular, it is straightforward to make the ZKRP schemes described in this document non-interactive using the Fiat–Shamir heuristic.

A zero knowledge proof scheme has the following properties:

- *Completeness* Given a witness w that satisfies instance x, we have that $\text{Verify}(\text{Prove}(x, w)) = 1$.

- *Soundness* If the witness w does not satisfy x, then the probability $\mathrm{Prob}[\mathrm{Verify}(\mathrm{Prove}(x, w)) = 1]$ is sufficiently low.
- *Zero Knowledge* Given the interaction between prover and verifier, we call this interaction a *view*. In order to capture the zero knowledge property we use a polynomial-time *simulator*, which has access to the same input given to the verifier (including its randomness), but no access to the input of the prover, to generate a *simulated view*. We say that the ZKP scheme has *perfect zero knowledge* if the simulated view, under the assumption that $x \in \mathcal{L}$, has the same distribution as the original view. We say that the ZKP scheme has *statistical zero knowledge* if those distributions are *statistically close*. We say that the ZKP scheme has *computational zero knowledge* if there is no polynomial-time distinguisher for those distributions. Intuitively, the existence of such a simulator means that whatever the verifier can compute from the interaction with the prover, it was already possible to compute before such interaction, hence the verifier learned nothing from it. Also, we say that it is a *proof of knowledge* if we can find an *extractor*, who has *rewindable* black-box access to the prover, that can compute the witness w with non-negligible probability.

3.4 Bilinear pairings

Some constructions of ZKRP are based on the existence of a secure bilinear map $\mathbf{bp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, g_1, g_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ are groups of sufficiently large prime order, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and e is an appropriate choice of bilinear map, satisfying the usual requirements: (i) non-degeneracy; (ii) efficiently computable and (iii) bilinearity. This cryptographic primitive is key to the constructions we will present in the next sections and it is important to remark that care must be taken when instantiating such primitive [32, 60]. Barreto-Naehrig [3] elliptic curves permit to implement bilinear maps efficiently.

4 Zero knowledge range proofs

The first constructions of ZKRP protocols were presented decades ago, with schemes like the one proposed in 1995 by Damgård [25] and in 1997 by Fujisaki and Okamoto [31]. Unfortunately those proposals are not efficient to be used in practice. The first practical construction was proposed by Boudot in 2001 [29]. In this document we will focus on constructions that came after Boudot's proposal.

In this section we describe in detail different strategies to achieve ZKRP. We can distinguish two different ways to commit to the secret: integer and binary. For each representation distinct strategies exist. A summary of the main characteristics of each family of constructions follows.

1. Integer representation proposals:
   - *Square decomposition* One of the ideas that can be used to obtain zero knowledge range proofs is the decomposition of the secret element into a sum of squares, as proposed in 2001 by Boudot [29]. In 2003 Lipmaa et al. [43] improved the construction using Lagrange's *four squares theorem*. In 2005 Groth [36] observed that if the element is in the form $4n + 1$, then it is possible to get the same result by decomposing only into three squares. The drawback of this approach is that the algorithm by Rabin and Shallit [53], required for the decomposition into squares, runs in time $\mathcal{O}(k^4)$, where k is the size of the secret. Both Lipmaa [43] and Groth [36] improved this algorithm, but in practice we have that it leads to a poor performance for the Prover's algorithm. In 2017, Couteau, Peters and Pointcheval showed how to remove the strong RSA assumption requirement [22] , providing an elegant description of previous schemes. In particular, the original constructions do not need to be modified in order to remove the strong RSA assumption. Also, they proposed a construction that allows faster verification and lower communication, at the price of a less efficient prover. This is good for DLT applications, because the verification in general must be executed by multiple parties and the proof must be stored in the ledger. Applications like anonymous credentials indeed require big secrets, and in this case the square decomposition is a good strategy to follow.
   - *Signature-based* Another idea for the prover is to prove, in a blind way, that he knows a signature on the secret. Initially, all elements in the interval are signed, then the proof that the prover knows the signature means that this integer belongs to the expected interval. In fact this interval can be any possible finite set, which means that this solution can be used to construct ZK Set Membership. In 2008 Camenisch, Chaabouni and shelat used bilinear pairings to construct an efficient ZKSM scheme [11] that may be used also for ZKRP. If the interval contains N numbers, this solution would require communication of $\mathcal{O}(N)$ digital signatures. The authors describe in the paper how to use u-ary representation to reduce the communication complexity to $\mathcal{O}(\frac{N}{\log N})$. In 2010, Chaabouni et al. improved the communication complexity by a factor of 2 [17].

2. Binary representation proposals:
   - *Multi-base decomposition* A common approach that one could follow to build ZKRP schemes is to decompose the secret into the bit representation, which allows to prove that it belongs to the interval by using Boolean arithmetic. Basically, the prover must commit to each bit of the secret; provide a zero knowledge proof that it is indeed a bit; and show a zero knowledge proof that the representation is valid. This last condition may easily be achieved by the utilization of homomorphic commitments. If instead of using the bit representation we use u-ary representation, then we can obtain more efficient constructions, as pointed out in [11]. Another possible strategy is to use the so-called *multi-base decomposition* [44, 56], which is an alternative way to represent the secret and it allows to build ZKRP schemes that are good for the case of small secrets.
   - *Two-tiered homomorphic commitments* [35] . In 2011 Groth proposed a new method to construct ZKRP which allows to obtain communication complexity $\mathcal{O}(N^{1/3})$, where N is the bit-length of the secret. Groth constructed an argument for batch multiplication of elements in $\mathbb{Z}_p$, which can be used to prove that $u_i.v_i = w_i$, where $u_i, v_i, w_i \in \mathbb{Z}_p$ for $0 \leq i < N$. To construct ZKRP, if the bits of the secret are given by w_i, then the argument can be used to show that $w_i.w_i = w_i$, what convinces the verifier that in fact $w_i \in \{0,1\}$. Also, he showed how to prove that $w = \sum_{i=0}^{N} w_i.2^i$, thus proving $w \in [0, 2^N)$. The argument can be easily adapted to a general interval $[A, B)$. The key idea of Groth's construction is to use bilinear pairings to commit to a vector of Pedersen commitments. For instance, given pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and elements $v, u_1, \dots, u_N \in \mathbb{G}_2$, we can commit to the vector $[c_1, \dots, c_N] \in \mathbb{G}_1^N$ by choosing a random $t \in \mathbb{G}$ and computing $C = e(t, v) \prod_{i=0}^{N} e(c_i, u_i)$.
   - *Bulletproofs* Unfortunately, all the schemes abovementioned depends upon a trusted setup, which may not be interesting in the context of cryptocurrencies. For instance, if an adversary is able to circumvent this trusted setup, he would be able to create money out of thin air. Recently, Bünz et al. [7] proposed a new idea to construct ZKRP, which they called *Bulletproofs*. They proposed to use an inner product proof in order to achieve ZKRP with very small proof sizes. Also, they showed how to use a component called *multi-exponentiation* in order to optimize their construction. The authors also provided an efficient implementation that shows their proposal is adequate for many practical scenarios. However, this proposal was not included in the comparison by Canard et al. [13], then one of the contributions of this work is to analyze how Bulletproofs compares to the other proposals.

4.1 Square decomposition construction

In this section we describe the algorithms necessary to implement the ZKRP proposed by Boudot [29] in 2001. This construction requires some building blocks, like the zero knowledge proof that two commitments hide the same secret and the zero knowledge proof that the secret is a square.

We denote the zero knowledge proof that two commitments hide the same secret by $\text{PK}_{\text{SS}} = \{x, r_1, r_2 : E = g_1^x h_1^{r_1} \wedge F = g_2^x h_2^{r_2}\}$. The parameters for the PK_{SS} scheme is given by $\text{params}_{\text{SS}} = (t, \ell, s_1, s_2)$, which must be set in order to achieve the desired level of security. Namely, we have that soundness is given by 2^{t-1}, while the zero knowledge property is guaranteed given that $1/\ell$ is negligible. Next we present algorithms Prove_{SS} and $\text{Verify}_{\text{SS}}$. It is important to remark that the discrete logarithm of g_1 with respect to h_1, or its inverse, must be unknown, otherwise the commitment is not secure. Analogously, we have that the same condition must be valid for g_2 and h_2. The hash function is such that it outputs $2t$-bit strings. Finally, we have that s_1 and s_2 must be chosen in order to have secure commitments, i.e. 2^{s_i} must be negligible for $i \in \{1, 2\}$.

Algorithm 1 Proof of Same Secret: Prove_{SS}

```
INPUT $x, r_1, r_2, E, F, \text{params}_{\text{SS}}$.
OUTPUT $\text{proof}_{\text{SS}}$.
  $\omega \in_R [1, 2^{\ell+t}b - 1]$,
  $\eta_1 \in_R [1, 2^{\ell+t+s_1}n - 1]$,
  $\eta_2 \in_R [1, 2^{\ell+t+s_2}n - 1]$,
  $\Omega_1 = g_1^{\omega} h_1^{\eta_1}$,
  $\Omega_2 = g_2^{\omega} h_2^{\eta_2}$,
  $c = \text{Hash}(\Omega_1 || \Omega_2)$,
  $D = \omega + cx$,
  $D_1 = \eta_1 + cr_1$,
  $D_2 = \eta_2 + cr_2$,
  return $\text{proof}_{\text{SS}} = (c, D, D_1, D_2)$.
```

Algorithm 2 Proof of Same Secret: $\text{Verify}_{\text{SS}}$

```
INPUT $E, F, \text{proof}_{\text{SS}}$.
OUTPUT True or false.
  return $c \overset{?}{=} \text{Hash}(g_1^D h_1^{D_1} E^{-c} || g_2^D h_2^{D_2} F^{-c})$.
```

We denote the zero knowledge proof that a secret is a square by $\text{PK}_{\text{S}} = \{x, r_1 : E = g^{x^2} h^r\}$. We have that $\text{params}_{\text{S}} = (t, \ell, s)$ represents the parameters for the PK_{S}

scheme, so that soundness is given by 2^{t-1} and the zero knowledge property is guaranteed if $1/\ell$ is negligible, as before. Algorithms 3 and 4 corresponds to Prove_S and Verify_S, respectively. Also, the discrete logarithm of g with respect to h, or its inverse, must be unknown, otherwise the commitment is not secure.

Algorithm 3 Proof of Square: Prove_S

```
INPUT $x, r_1, E, \text{params}_\text{S}$.
OUTPUT $\text{proof}_\text{S}$.
  $r_2 \in_R [-2^s n + 1, 2^s n - 1]$,
  $F = g^x h^{r_2}$,
  $r_3 = r_1 - r_2 x$,
  $\text{proof}_S S = \text{Prove}_S S(x, r_2, r_3, E, F)$,
  return $\text{proof}_S = (E, F, \text{proof}_{\text{SS}})$.
```

Algorithm 4 Proof of Square: Verify_S

```
INPUT $\text{proof}_\text{S}$.
OUTPUT True or false.
  return $\text{Verify}_{\text{SS}}(E, F, \text{proof}_{\text{SS}})$.
```

We denote the zero knowledge proof that a secret belongs to a larger interval, originally proposed by Chan et al. [19], by using notation $\text{PK}_{\text{LI}} = \{x, r : E = g^x h^r \wedge x \in [-2^{t+\ell} b, 2^{t+\ell} b]\}$. We have that $\text{params}_{\text{LI}} = (t, \ell, s)$ represents the parameters for the PK_{LI} scheme, so that completeness is achieved with probability greater than $1 - 2^\ell$; soundness is given by 2^{t-1} and the zero knowledge property is guaranteed if $1/\ell$ is negligible. Algorithms 5 and 6 corresponds to Prove_{LI} and $\text{Verify}_{\text{LI}}$, respectively. Also, the discrete logarithm of g with respect to h, or its inverse, must be unknown.

Algorithm 5 Proof of Larger Interval: Prove_{LI}

```
INPUT $x, r, E, \text{params}_{\text{LI}}$.
OUTPUT $\text{proof}_{\text{LI}}$.
  repeat
    $\omega \in_R [0, 2^{t+\ell} b - 1]$,
    $\eta \in_r [-2^{t+\ell+s} n + 1, 2^{t+\ell+s} n - 1]$,
    $\Omega = g^\omega h^\eta \pmod{n}$,
    $C = \text{Hash}(\Omega)$,
    $c = C \pmod{2^t}$,
    $D_1 = \omega + xc$,
    $D_2 = \eta + xc \in \mathbb{Z}$,
  till $D_1 \in [cb, 2^{t+l} b - 1]$.
  return $(C, D_1, D_2)$.
```

Algorithm 6 Proof of Larger Interval: $\text{Verify}_{\text{LI}}$

```
INPUT $\text{proof}_{\text{LI}}$.
OUTPUT True or false.
  return $D_1 \overset{?}{\in} [cb, 2^{t+\ell} b - 1] \wedge C \overset{?}{=} \text{Hash}(g^{D_1} h^{D_2} E^{-c})$.
```

Before describing Boudot's ZKRP construction, we first need a *proof with tolerance*, denoted by $\text{PK}_{\text{WT}} = \{x, r : E = g^x h^r \wedge x \in [a - \theta, b + \theta]\}$, where $\theta = 2^{t+\ell+1}\sqrt{b-a}$, as shown in Algorithms 7 and 8.

Algorithm 7 Proof with Tolerance: Prove_{WT}

```
INPUT $x, r, E$.
OUTPUT $\text{proof}_{\text{WT}}$.
  Compute the proof of opening of commitment $E$.
  $E_a = E/g^a \pmod{n}$,
  $E_b = g^b/E \pmod{n}$,
  $x_a = x - a$,
  $x_b = b - x$,
  Alice proves that she knows $x$, which is greater that $-\theta$.
  $x_{a_1} = \lfloor\sqrt{x-a}\rfloor$,
  $x_{a_2} = x_a - x_{a_1}^2$,
  $x_{b_1} = \lfloor\sqrt{b-x}\rfloor$,
  $x_{b_2} = x_b - x_{b_1}^2$,
  repeat
    $r_{a_1} \in_R [-2^s n + 1, 2^s n - 1]$,
    $r_{a_2} = r - r_{a_1}$,
  till $r_{a_2} \in [-2^s n + 1, 2^s n - 1]$.
  Choose $r_{b_1}$ and $r_{b_2}$ such that $r_{b_1} + r_{b_2} = -r$.
  $E_{a_1} = g^{x_{a_1}^2} h^{r_{a_1}}$,
  $E_{a_2} = g^{x_{a_2}} h^{r_{a_2}}$,
  $E_{b_1} = g^{x_{b_1}^2} h^{r_{b_1}}$,
  $E_{b_2} = g^{x_{b_2}} h^{r_{b_2}}$,
  $\text{proof}_{\text{S}_a} = \text{Prove}_\text{S}(x_{a_1}, r_{a_1}, E_{a_1})$,
  $\text{proof}_{\text{S}_b} = \text{Prove}_\text{S}(x_{b_1}, r_{b_1}, E_{b_1})$,
  $\text{proof}_{\text{LI}_a} = \text{Prove}_{\text{LI}}(x_{a_2}, r_{a_2}, E_{a_2})$,
  $\text{proof}_{\text{LI}_b} = \text{Prove}_{\text{LI}}(x_{b_2}, r_{b_2}, E_{b_2})$,
  return $\text{proof}_{\text{WT}} = (E_{a_1}, E_{a_2}, E_{b_1}, E_{b_2}, \text{proof}_{\text{S}_a}, \text{proof}_{\text{S}_b}, \text{proof}_{\text{LI}_a}, \text{proof}_{\text{LI}_b})$.
```

Algorithm 8 Proof with Tolerance: $\text{Verify}_{\text{WT}}$

```
INPUT $\text{proof}_{\text{WT}}$.
OUTPUT True or false.
  if $E_{a_2} \overset{?}{=} E_a/E_{a_1} \wedge E_{b_2} \overset{?}{=} E_b/E_{b_1}$ then
    $b_\text{S} = \text{Verify}_\text{S}(\text{proof}_{\text{S}_a}) \wedge \text{Verify}_\text{S}(\text{proof}_{\text{S}_b})$,
    $b_{\text{LI}} = \text{Verify}_{\text{LI}}(\text{proof}_{\text{LI}_a}), \wedge \text{Verify}_{\text{LI}}(\text{proof}_{\text{LI}_b})$,
    return $b_\text{S} \wedge b_{\text{LI}}$.
  return False
```

Algorithms 9 and 10 describe the ZKRP scheme proposed by Boudot [29] in 2001.

Algorithm 9 Square Decomposition Range Proof: Prove_{SD}

```
INPUT $x, r, R$.
OUTPUT $\text{proof}_{\text{SD}}$.
  $x' = 2^T x$,
  $r' = 2^T r$,
  $T = 2(t + \ell + 1) + |b - a|$,
  $E' = E^{2^T}$,
  $\text{proof}_{\text{WT}} = \text{Prove}_{\text{WT}}(x', r', E')$,
  return $\text{proof}_{\text{SD}} = (E', \text{proof}_{\text{WT}})$.
```

Algorithm 10 Square Decomposition Range Proof: $\text{Verify}_{\text{SD}}$

```
INPUT $\text{proof}_{\text{SD}}$.
OUTPUT True or false.
  if $E' \overset{?}{=} E^{2^T}$ then
    return $\text{Verify}_{\text{WT}}(\text{proof}_{\text{WT}})$.
  return False
```

4.2 Signature-based construction

The idea of the protocol is that the verifier initially computes digital signatures for each element in the target set S. The prover then blinds this digital signature by raising it to a randomly chosen exponent $v \in \mathbb{Z}_p$, such that it is computationally infeasible to determine which element was signed. The prover uses the pairing to compute the proof, and the bilinearity of the pairing allows the verifier to check that indeed one of the elements from S were initially chosen. Algorithms 11, 12 and 13 show the details of the this protocol. The scheme depends upon Boneh-Boyen digital signatures, summarized in next.

Boneh-Boyen [27] *signatures* Shortly, the signer private key is given by $x \in_R \mathbb{Z}_p$ and the public key is $y = g^x$. Given message m, we have that the digital signature is calculated as $\sigma = g^{1/(x+m)}$, and verification is achieved by computing $e(\sigma, yg^m) \overset{?}{=} e(g, g)$.

Boneh-Boyen signatures are based on the q-Strong Diffie-Hellman assumption, described in Definition 4.

Algorithm 11 Set Membership: $\text{Setup}_{\text{ZKSM}}$

```
INPUT $g, h$ and a set $S$.
OUTPUT $y \in \mathbb{G}$ and $A \in G^{|S|}$.
  $x \in_R \mathbb{Z}_p$,
  $y = g^x$ ,
  for $i \in S$ do
    $A_i = g^{\frac{1}{x+i}}$.
  return $y, [A_i]$.
```

Algorithm 12 Set Membership: $\text{Prove}_{\text{ZKSM}}$

```
INPUT $g, h$, a commitment $C$, and a set $S$.
OUTPUT $\delta, \gamma$ such that $C = g^\delta h^\gamma$ and $\delta \in S$.
  $\tau \in_R \mathbb{Z}_p$,
  $V = A_\delta^\tau$,
  $s, t, m \in_R \mathbb{Z}_p$,
  $a = e(V, g)^{-s}.e(g, g)^t$,
  $D = g^s h^m$,
  $c = \text{Hash}(V, a, D)$,
  $z_\delta = s - \delta c$,
  $z_\tau = t - \tau c$,
  $z_\gamma = m - \gamma c$.
  return $\text{proof}_{\text{ZKSM}} = (V, a, D, z_\delta, z_\tau, z_\gamma)$.
```

Algorithm 13 Set Membership: $\text{Verify}_{\text{ZKSM}}$

```
INPUT $g, h$, a commitment $C$, $\text{proof}_{\text{ZKSM}}$.
OUTPUT True or false.
  return $D \overset{?}{=} C^c h^{z_\gamma} g^{z_\delta} \wedge a \overset{?}{=} e(V, y)^c.e(V, g)^{-z_\delta}.e(g, g)^{z_\tau}$.
```

Range Proof In order to obtain ZKRP, we can decompose the secret δ into base u, as follows:

$$\delta = \sum_{0 \le j \le \ell} \delta_j u^j.$$

Therefore, if each δ_j belongs to the interval $[0, u)$, then we have that $\delta \in [0, u^\ell)$. The ZKSM algorithms can be easily adapted to carry out this computation, as shown in Algorithms 14, 15 and 16.

Algorithm 14 Signature-based Range Proof: $\text{Setup}_{\text{ZKRP}}$ for interval $[0, u^\ell)$

```
INPUT $g, h, u, \ell$ and a commitment $C$.
OUTPUT $\delta, \gamma$ such that $C = g^\delta h^\gamma$ and $\delta \in [0, u^\ell)$.
  $x \in_R \mathbb{Z}_p$
  $y = g^x$
  for $i \in \mathbb{Z}_u$ do
    $A_i = g^{\frac{1}{x+i}}$.
  return $y, [A_i]$.
```

Algorithm 15 Signature-based Range Proof: $\text{Prove}_{\text{ZKRP}}$ for interval $[0, u^\ell)$

```
INPUT $g, h, u, \ell$ and a commitment $C$.
OUTPUT $\delta, \gamma$ such that $C = g^\delta h^\gamma$ and $\delta \in [0, u^\ell)$.
  Find $[\delta_j]$ such that $\delta = \sum_j \delta_j u^j$,
  $\tau_j \in_R \mathbb{Z}_p$,
  Set $D$ to the identity element in $\mathbb{G}$.
  for $j \in \mathbb{Z}_\ell$ do
    $V_j = A_{\delta_j}^{\tau_j}$,
    $s_j, t_j, m_j \in_r \mathbb{Z}_p$,
    $a_j = e(V_j, g)^{-s_j}.e(g, g)^{t_j}$,
    $D = Dg^{u^j s_j} h^{m_j}$.
  $c = \text{Hash}([V_j], a, D)$.
  for $j \in \mathbb{Z}_\ell$ do
    $z_{\delta_j} = s_j - \delta_j c$,
    $z_{\tau_j} = t_j - \tau_j c$.
  $z_\gamma = m - \gamma c$.
  return $\text{proof}_{\text{ZKRP}} = (z_\gamma, [z_{\delta_j}], [z_{\tau_j}])$.
```

Algorithm 16 Signature-based Range Proof: $\text{Verify}_{\text{ZKRP}}$ for interval $[0, u^\ell)$

```
INPUT $g, h, u, \ell$ and $\text{proof}_{\text{ZKRP}}$.
OUTPUT True or false.
  Set $a$ to True.
  for $j \in \mathbb{Z}_\ell$ do
    $a = a \wedge (a_j \overset{?}{=} e(V_j, y)^c.e(V_j, g)^{-z_{\delta_j}}.e(g, g)^{z_{\tau_j}})$.
  return $D \overset{?}{=} C^c h^{z_\gamma} \prod_j (u^j z_{\delta_j}) \wedge a$.
```

In order to obtain Zero Knowledge Range Proofs for arbitrary ranges $[a, b)$ we show that $\delta \in [a, a + u^\ell)$ and $\delta \in [b - u^\ell, b)$, using 2 times the ZKRP scheme described in Algorithm 15. Namely, we have to prove that $\delta - b + u^\ell \in [0, u^\ell)$ and $\delta - a \in [0, u^\ell)$.

4.3 Bulletproofs construction

In this section we show a detailed description of the algorithms necessary to implement the Bulletproofs ZKRP protocol.

Notation Given an array $\mathbf{a} \in \mathbb{G}^n$, we use Python notation to represent array slices:

$$\mathbf{a}_{[:\ell]} = [a_1, \dots, a_\ell] \in \mathbb{G}^\ell,$$
$$\mathbf{a}_{[\ell:]} = [a_{\ell+1}, \dots, a_n] \in \mathbb{G}^{n-\ell}$$

Given $k \in \mathbb{G}$, we denote the vector containing the powers of k by

$$\mathbf{k}^n = [1, k, k^2, \dots, k^{n-1}].$$

Given $\mathbf{g} = [g_1, \dots, g_n] \in \mathbb{G}^n$ and $\mathbf{a} \in \mathbb{Z}_p^n$, we define $\mathbf{g}^\mathbf{a}$ as follows:

$$\mathbf{g}^\mathbf{a} = \prod_{i=1}^n g_i^{a_i}.$$

Given $c \in \mathbb{Z}_p$, notation $\mathbf{b} = c.\mathbf{a} \in \mathbb{Z}_p^n$ is a vector such that $b_i = c.a_i$. Also, $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \dots, a_n b_n)$ is the Hadamard product. The vector polynomial $p(X) = \sum_{i=0}^n \mathbf{p}_i X^i \in \mathbb{Z}_p^n[X]$, where each coefficient $\mathbf{p}_i$ is a vector in $\mathbb{Z}_p^n$. The inner product of such polynomials is given by

$$\langle \mathbf{l}(X), \mathbf{r}(X) \rangle = \sum_{i=0}^d \sum_{j=0}^i \langle \mathbf{l}_i, \mathbf{r}_j \rangle X^{i+j} \in \mathbb{Z}_p[X]. \quad (1)$$

4.3.1 Setup

Many ZKRP constructions depend on a trusted setup. Shortly, the parameters necessary to generate and verify the underlying zero knowledge proofs must be computed by a trusted party, because if such parameters are generated using a *trapdoor*, then this trapdoor could be used to subvert the protocol, allowing to generate money out of thin air.

In order to avoid the trusted setup, Bulletproofs use the *Nothing Up My Sleeve* (NUMS) strategy, where a hash function [6] is utilized to compute the generators that will be necessary for the Pedersen commitments, as described in Algorithm 17, which describes the specific case where the subjacent elliptic curve is given by Koblitz curve `secp256k1` [15, 38].

Algorithm 17 Nothing Up My Sleeve: MapToGroup

```
INPUT The input string $m$, and the field prime modulus $p$.
OUTPUT An elliptic curve point if successful or some error.
  $i = 0$.
  while $i < 256$ do
    $x = \text{Hash}(m, i)$.
    $\text{rhs} = x^3 + 7 \pmod p$.
    if rhs is a square $\pmod p$ then
      $y = \sqrt{\text{rhs}} \pmod p$.
      if $(x, y)$ is not the point at infinity then
        return $(x, y)$.
    $i = i + 1$.
  return "Can not map to group".
```

```
Algorithm 18 Compute Generators: ComputeGenerators
INPUT The elliptic curve public generator $g \in \mathbb{G}$ and an integer $n$.
OUTPUT The set of generators $(g, h, \mathbf{g}, \mathbf{h})$.
  Compute $h = \text{MapToGroup}('\text{some public string}', p)$.
  $i = 0$.
  while $i < n$ do
    $c \in_R \mathbb{Z}_p$,
    $d \in_R \mathbb{Z}_p$,
    $\mathbf{g}[i] = c.G$,
    $\mathbf{h}[i] = d.G$,
    $i = i + 1$.
  return $(g, h, \mathbf{g}, \mathbf{h})$.
```

```
Algorithm 19 Setup_IP
INPUT The set of generators $(g, h, \mathbf{g}, \mathbf{h})$.
OUTPUT $\text{params}_{\text{IP}}$.
  $u = \text{MapToGroup}('\text{some other public string}', p)$.
  return $\text{params}_{\text{IP}} = (g, h, \mathbf{g}, \mathbf{h}, u)$.
```

```
Algorithm 20 Setup_RP
INPUT The input interval $[a, b)$ and the field modulus $p$.
OUTPUT $\text{params}_{\text{RP}}$.
  if $b$ is not a power of 2 then
    return "$b$ must be a power of 2".
  else
    $n = \log_2 b$,
    $(g, h, \mathbf{g}, \mathbf{h}) = \text{ComputeGenerators}(g, n)$,
    $\text{params}_{\text{IP}} = \text{Setup}_{\text{IP}}(g, h, \mathbf{g}, \mathbf{h})$,
    $\text{params}_{\text{RP}} = (\text{params}_{\text{IP}}, n)$,
    return $\text{params}_{\text{RP}}$.
```

4.3.2 Inner product argument

In this section we present the main building block of Bulletproofs, which is the inner product argument. In summary, using this ZKP protocol the prover convinces a verifier that she knows vectors whose inner product is equal to a determined public value. First we describe the initialization procedure in Algorithm 22. Afterwards we present the main protocol, given by Algorithm 23.

```
Algorithm 21 Vector Commitment: Commit_IP
INPUT $(\text{params}_{\text{IP}}, \mathbf{a}, \mathbf{b})$.
OUTPUT The commitment $P$.
  Compute $P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \in \mathbb{G}$.
  return $P$.
```

```
Algorithm 22 Proof of Inner Product: Prove_IP
INPUT $(\text{params}_{\text{IP}}, \text{commit}_{\text{IP}}, c, \mathbf{a}, \mathbf{b})$.
OUTPUT $\text{proof}_{\text{IP}}$.
  $x = \text{Hash}(\mathbf{g}, \mathbf{h}, P, c) \in \mathbb{Z}_p^\star$
  Compute $P' = u^{x \cdot c}P$.
  Allocate arrays $\mathbf{l}, \mathbf{r} \in \mathbb{G}^n$.
  $\text{ComputeProof}(\mathbf{g}, \mathbf{h}, P', u^x, \mathbf{a}, \mathbf{b}, \mathbf{l}, \mathbf{r})$.
  $\text{proof}_{\text{IP}} = (\mathbf{g}, \mathbf{h}, P', u^x, \mathbf{a}, \mathbf{b}, \mathbf{l}, \mathbf{r})$.
  return $\text{proof}_{\text{IP}}$.
```

```
Algorithm 23 Proof of Inner Product: ComputeProof
INPUT $(\mathbf{g}, \mathbf{h}, P, u, a, b, \mathbf{l}, \mathbf{r})$.
OUTPUT $(\mathbf{g}, \mathbf{h}, P, u, a, b, \mathbf{l}, \mathbf{r})$.
  $x = \text{Hash}(\mathbf{g}, \mathbf{h}, P, c) \in \mathbb{Z}_p^\star$.
  Compute $P' = u^{x \cdot c}P$.
  if $n = 1$ then
    return $(\mathbf{g}, \mathbf{h}, P, u, a, b, \mathbf{l}, \mathbf{r})$.
  else
    $n' = \frac{n}{2}$,
    $c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p$,
    $c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p$,
    $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} u^{c_L} \in \mathbb{G}$,
    $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} u^{c_R} \in \mathbb{G}$.
    Append $L, R$ to $\mathbf{l}, \mathbf{r}$, respectively.
    $x = \text{Hash}(L, R)$,
    $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$,
    $\mathbf{h}' = \mathbf{g}_{[:n']}^{x} \mathbf{g}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'}$,
    $P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$,
    $\mathbf{a}' = \mathbf{a}_{[:n']}x + \mathbf{a}_{[n':]}x^{-1} \in \mathbb{Z}_p^{n'}$,
    $\mathbf{b}' = \mathbf{b}_{[:n']}x^{-1} + \mathbf{b}_{[n':]}x \in \mathbb{Z}_p^{n'}$.
    Recursively run ComputeProof on input $(\mathbf{g}', \mathbf{h}', P', u, \mathbf{a}', \mathbf{b}', \mathbf{l}, \mathbf{r})$.
```

```
Algorithm 24 Proof of Inner Product: Verify_IP
INPUT $\text{params}_{\text{IP}}, \text{commit}_{\text{IP}}, \text{proof}_{\text{IP}}$.
OUTPUT True or false.
  $i = 0$.
  while $i < \log n$ do
    $n' = \frac{n}{2}$,
    $x = \text{Hash}(\mathbf{l}[i], \mathbf{r}[i])$,
    $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$,
    $\mathbf{h}' = \mathbf{g}_{[:n']}^{x} \mathbf{g}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'}$,
    $P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$,
    $i = i + 1$.
  The verifier computes $c = a.b$ and accepts if $P = g^a h^b u^c$.
```

The fact that Bulletproofs allows to halve the size of the problem in each level of the recursion in Algorithm 23 means that it is possible to obtain logarithmic proof size.

4.3.3 Range proof argument

Given a secret value v, if we want to prove it belongs to the interval $[0, 2^n)$, then we do the following:

- Prove that $\mathbf{a}_L \in \{0,1\}^n$ is the bit-decomposition of v. In other words, we show that

  $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v.$
- Define $\mathbf{a}_R$ as the component-wise complement of $\mathbf{a}_L$, what means that, for every $i \in [0, n]$, if the i-th bit of $\mathbf{a}_L$ is 0, then the i-th bit of $\mathbf{a}_R$ is equal to 1. Conversely, if the i-th bit of $\mathbf{a}_L$ is 1, then the i-th bit of $\mathbf{a}_R$ is equal to 0. Equivalently, this condition can be shortly described by Eqs. 2 and 3.

$$\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n, \tag{2}$$

$$\mathbf{a}_R = \mathbf{a}_L - 1^n \pmod 2). \tag{3}$$

In order to prove that $\mathbf{a}_L$ and $\mathbf{a}_R$ satisfy both relations, we can randomly choose $y \in \mathbb{Z}_p$ and compute:

$$\langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle = 0,$$
$$\langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n \rangle = 0.$$

These two equations can be combined into a single inner product, by randomly choosing $z \in \mathbb{Z}_p$, and computing

$$\langle \mathbf{a}_L - z.\mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z.\mathbf{1}^n) + z^2.\mathbf{2}^n \rangle = z^2 v + \delta(y,z), \tag{4}$$

where $\delta(y,z) = (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle \in \mathbb{Z}_p$.

If the prover could send the vectors in Eq. 4, then the verifier would be able to check the inner product himself. However, this vector reveals information about $\mathbf{a}_L$, therefore revealing bits of the secret value v. To solve this problem the prover randomly chooses vectors $\mathbf{s}_L$ and $\mathbf{s}_R$ in order to blind $\mathbf{a}_L$ and $\mathbf{a}_R$, respectively. Consider the following polynomials:

$$l[X] = \mathbf{a}_L - z.1^n + s_L.X \in \mathbb{Z}_p^n,$$
$$r[X] = \mathbf{y}^n \circ (\mathbf{a}_R + z.1^n + \mathbf{s}_R.X) + z^2 2^n \in \mathbb{Z}_p^n,$$
$$t[X] = \langle l[X], r[X] \rangle = t_0 + t_1.X + t_2.X^2,$$

where the above inner product is computed as defined in Eq. 1.

Note that the constant terms of $l[X]$ and $r[X]$ correspond to the vectors in Eq. 4. Therefore if the prover publishes $l[x]$ and $r[x]$ for a specific $x \in \mathbb{Z}_p$, then we have that terms $\mathbf{s}_L$ and $\mathbf{s}_R$ ensure no information about $\mathbf{a}_L$ and $\mathbf{a}_R$ is revealed.

Explicitly, we have that

$$t_1 = \langle \mathbf{a}_L - z.\mathbf{1}^n, \mathbf{y}^n.\mathbf{s}_R \rangle + \langle \mathbf{s}_L, \mathbf{y}^n.(\mathbf{a}_R + z.\mathbf{1}^n) \rangle, \tag{5}$$

and

$$t_2 = \langle \mathbf{s}_L, \mathbf{y}^n.\mathbf{s}_R \rangle. \tag{6}$$

Algorithm 25 Bulletproofs: Prove$_{\text{RP}}$

```
INPUT params_RP, v.
OUTPUT proof_RP.
  $\gamma \in_R \mathbb{Z}_p$,
  $V = g^v h^\gamma \in \mathbb{G}$,
  $\mathbf{a}_L \in \{0,1\}^n$ such that $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$,
  $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \in \mathbb{Z}_p^n$,
  $\alpha \in_R \mathbb{Z}_p$,
  $A = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$,
  $s_L, s_R \in_R \mathbb{Z}_p^n$,
  $\rho \in_R \mathbb{Z}_p$,
  $S = h^\rho \mathbf{g}^{s_L} \mathbf{h}^{s_R} \in \mathbb{G}$,
  $y = \mathrm{Hash}(A,S) \in \mathbb{Z}_p^\star$,
  $z = \mathrm{Hash}(A,S,y) \in \mathbb{Z}_p^\star$,
  $\tau_1, \tau_2 \in_R \mathbb{Z}_p$,
  $T_1 = g^{t_1} h^{\tau_1} \in \mathbb{G}$,
  $T_2 = g^{t_2} h^{\tau_2} \in \mathbb{G}$,
  $x = \mathrm{Hash}(T_1,T_2) \in \mathbb{Z}_p^\star$,
  $\mathbf{l} = l(X) = \mathbf{a}_L - z1^n + s_L X \in \mathbb{Z}_p^n$,
  $\mathbf{r} = r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z1^n + \mathbf{s}_R X) + z^2 2^n \in \mathbb{Z}_p^n$,
  $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$,
  $\tau_x = \tau_2 x^2 + \tau_1 x + z^2 \gamma \in \mathbb{Z}_p$,
  $\mu = \alpha + \rho x \in \mathbb{Z}_p$,
  $\mathrm{commit}_{\mathrm{IP}} = \mathrm{Commit}_{\mathrm{IP}}(\mathrm{params}_{\mathrm{IP}}, \mathbf{l}, \mathbf{r})$,
  $\mathrm{proof}_{\mathrm{IP}} = \mathrm{Prove}_{\mathrm{IP}}(\mathrm{params}_{\mathrm{IP}}, \mathrm{commit}_{\mathrm{IP}}, \hat{t}, \mathbf{l}, \mathbf{r})$,
  $\mathrm{proof}_{\mathrm{RP}} = (\tau_x, \mu, \hat{t}, V, A, S, T_1, T_2, \mathrm{commit}_{\mathrm{IP}}, \mathrm{proof}_{\mathrm{IP}})$.
  return proof_RP.
```

Algorithm 26 Bulletproofs: Verify$_{\text{RP}}$

```
INPUT params_RP, proof_RP.
OUTPUT True or false.
  $y = \mathrm{Hash}(A,S) \in \mathbb{Z}_p^\star$,
  $z = \mathrm{Hash}(A,S,y) \in \mathbb{Z}_p^\star$,
  $x = \mathrm{Hash}(T_1,T_2) \in \mathbb{Z}_p^\star$,
  $h_i = h_i^{y^{-i+1}} \in \mathbb{G}, \forall i \in [1,n]$,
  $P_l = P.h^\mu$,
  $P_r = A.S^x.\mathbf{g}^{-z}.(\mathbf{h}')^{z.\mathbf{y}^n + z^2.\mathbf{2}^n} \in \mathbb{G}$,
  $\mathrm{output}_1 = (P_l \overset{?}{=} P_r)$,
  $\mathrm{output}_2 = (g^{\hat{t}} h^{\tau_x} \overset{?}{=} V^{z^2}.g^{\delta(y,z)}.T_1^x.T_2^{x^2})$,
  $\mathrm{output}_3 = \mathrm{Verify}_{\mathrm{IP}}(\mathrm{proof}_{\mathrm{IP}})$,
  return $\mathrm{output}_1 \wedge \mathrm{output}_2 \wedge \mathrm{output}_3$.
```

In order to make Bulletproofs non-interactive using the Fiat–Shamir heuristic. Concretely, we compute $x = \mathrm{Hash}(T_1, T_2)$, $y = \mathrm{Hash}(A, S)$, and $z = \mathrm{Hash}(A, S, y)$ in Algorithms 25 and 26.

4.3.4 Optimizations

The algorithms described in last section can be optimized in two ways, as follows:

- *Multi-exponentiation* In the inner-product argument presented in Section 4.3.2 it is required to computed many exponentiations, which is an expensive operation. For instance, in the k-th round of the protocol we must perform $\frac{n}{2^{k-1}}$ exponentiations, thus in total we must execute $4n$ exponentiations. It is possible to reduce this number to a single *multi-exponentiation* of size $2n$ by postponing these computations to the last round.

  Concretely, given $\mathbf{g} = [g_1, \dots, g_n]$, we have that it is possible to compute g an h, the generators obtained in last round, by using the following expressions:

$$g = \prod_{i=1}^{n} g_i^{s_i} \in \mathbb{G},$$

$$h = \prod_{i=1}^{n} h_i^{1/s_i} \in \mathbb{G},$$

  where

$$s_i = \prod_{j=1}^{\log_2 n} x_j^{b(i,j)}$$

  and

$$b(i,j) = \begin{cases} 1, & \text{if the } j-\text{th bit of} i-1 \text{ is } 1 \\ -1, & \text{otherwise} \end{cases}$$

  Therefore, verification can be performed by

$$\mathbf{g}^{a.s} \cdot \mathbf{h}^{b.s^{-1}} \cdot u^{a.b} \overset{?}{=} P \cdot \prod_{j=1}^{\log_2 n} L_j^{x_j^2} \cdot R_j^{x_j^{-2}}$$

- *Aggregation* If multiple range proofs use the same underlying interval, then it is possible to aggregate them into one single ZKRP. Using this optimization, we have that new proofs can be added by only increasing the total size of the proof by a logarithmic factor. Consider we want to aggregate m range proofs. Then, while the naive strategy would lead us to a proof whose size is m times larger, this aggregation procedure in Bulletproofs allows the proof to grow only by a factor of $2 \log_2 m$.

  In practice, applications like Confidential Transactions [46], Mimblewimble [52] and Provisions [23] would benefit a lot from the utilization of aggregation, because indeed such applications must execute many ZKRPs over the same interval.

Table 1 Time complexity

Scheme	Setup (ms)	Prove (ms)	Verify (ms)
[29]	331.41	579.32	851.89
[11]	31.78	70.18	98.95
[7]	13.11	96.25	51.86
[7][a]	17.20	22.38	3.27

[a]Optimized implementation

5 Implementation

We implemented the constructions described in Sects. 4.1, 4.2 and 4.3. The scheme based on square decomposition, i.e. Boudot's construction, was implemented in Java and Solidity, while the signature-based scheme and Bulletproofs were implemented in Golang and they were based on libsecp256k1 library, available in Go-Ethereum. We also provide an implementation of the verification algorithm for Bulletproofs in Solidity. We used BN128 pairing-friendly elliptic curves, thus accomplishing 128 bits of security. The performance is summarized in Table 1, and the measurement was carried out in a computer with a 64-bit Intel i5-6300U 2.40 GHz CPU, 16 GB of RAM and Ubuntu 18.04. The implementation is available on Github [49] and is a proof of concept, thus it should not be used in production without first spending the effort to review it where necessary.

Optimal values for u and ℓ can be calculated as described in the original paper [11]. We used $u = 57$ and $\ell = 5$ for the interval [347, 184, 000, 599, 644, 800), obtaining communication complexity equal to 30,976 bits, while the previous work, based on Boudot's proposal [29], has 48,946 bits.

Although schemes were implemented in different languages, Table 1 allows to understand that in spite of the asymptotic complexities of each proposal, the practical results show that schemes based on square decomposition are one order of magnitude slower than other implementations, due mainly to the fact that they involve large variables. Also, it is possible to see that Bulletproofs optimizations, represented in last row, are important to significantly reduce the verification time.

A qualitative comparison is presented in next section, showing the expected performance for different range sizes.

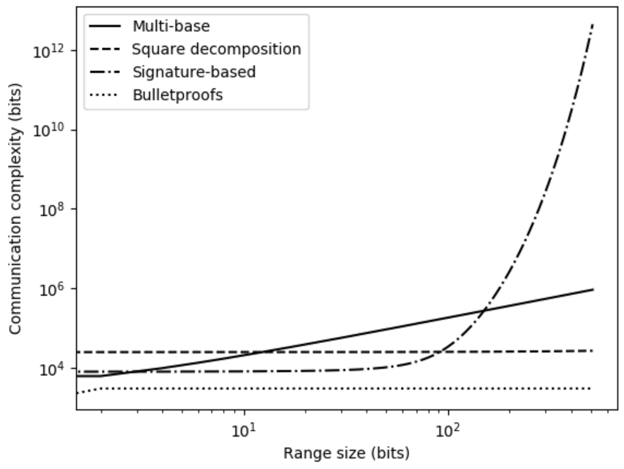

Fig. 1 Proofs size

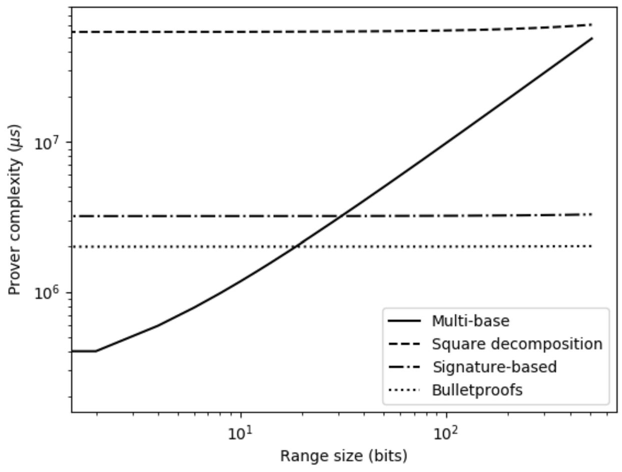

Fig. 2 Prover complexity

6 Comparison

In this section we compare the representative schemes for different strategies, as presented in Section 4, with respect to proof size and the complexity of Prove and Verify algorithms. It is important to emphasize that in this section we used different schemes than the ones we implemented, because we followed the qualitative comparison provided in [13]. However the schemes that follow the same strategy have similar performance, what allows us to have a good overview of the existing constructions. Namely, Boudot's proposal [29] have very similar performance when compared to Lipmaa's [43] and Groth's [36] constructions. For instance, we used the proposal by Lipmaa et al. [44] to represent the multi-base solution; the proposal by Lipmaa [43] to represent the square decomposition strategy; the scheme by Camenisch et al. [11] for the signature-based implementation; and we included Bulletproofs construction.

Compared to other proposals in the literature, we found that for very big intervals, the best strategy is to use the square decomposition, as for example occurs in the construction by Boudot [29], since verification doesn't depend on the size of the secret. However, it is important to remark that finding the decomposition into squares consumes a reasonable amount of computational resources, what makes the Prover's algorithm somewhat inefficient. This scenario may arise in the context of anonymous credentials [20, 24, 45]. On the other hand, for small secrets, Schoenmakers's strategy [57] is the most efficient scheme with respect to Prove algorithm.

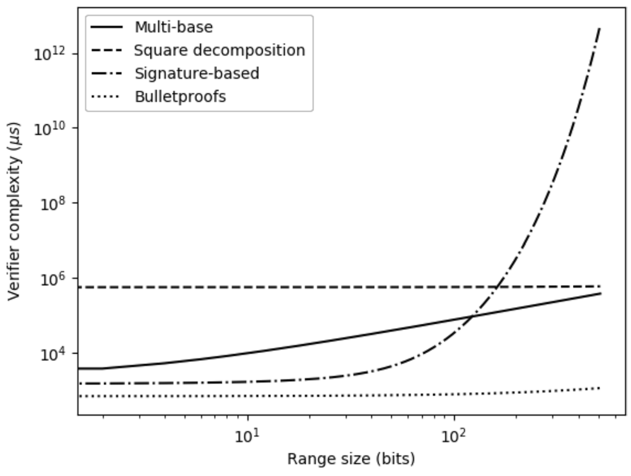

Fig. 3 Verifier complexity

In Figures 1, 2 and 3 we represent in the horizontal axis the bit-length of b, where b is the largest element from the subjacent range $[a, b]$ used for the zero knowledge range proof scheme.

It is possible to conclude that in general Bulletproofs offers the best performance, but depending on the requirements of the underlying chosen use case, it may be possible that other strategies offer better advantages. For DLT applications we have that the proof size and the verifier's complexity are more important metrics than the prover's complexity, what means that indeed Bulletproofs seems to be the best approach to implement a ZKRP protocol.

7 Related work and final remarks

In this document we described in detail the construction of ZKRP protocols, which were implemented and open-sourced. Most works on this subject are devoted to construct *non-interactive* protocols by using the Fiat–Shamir heuristic. Those constructions therefore rely on the random oracle model [4], which is considered a weaker model, since there are schemes proven secure in this model, but in practice turn out to be insecure. Chaabouni et al. [18] proposed a solution to this problem, allowing to construct non-interactive ZKRP in the standard model.

The focus of this work was on ZKRP, which is closely related to Zero Knowledge Set Membership protocols. More information on this topic can be found in Chaabouni's Ph.D. thesis [16].

Another related topic is called *cryptographic accumulator* [2, 8, 12]. It not only allows to verify membership in a set, but also permit to dynamically add and remove elements from the set. Accumulators can be used in replacement for the ZK Set Membership, thus can be a building block to construct ZK Range Proofs, as pointed out in [11]. Recently, Bünz et al. [5] used groups of unknown order to construct accumulators that allows batching. The technique is a generalization of the proof of exponent by Schnorr [55], and can be used to prove knowledge of a homomorphism pre-image. This work allows to reduce the huge amount of memory that is necessary to validate transactions in Bitcoin. In summary, the accumulator is used to store *unspent transaction outputs* (UTXOs), which can be added and removed using only constant-size memory. Also, the underlying digital signature scheme used in signature-based ZKRP [11], namely Boneh-Boyen signatures, can be replaced and the construction presented here can be adapted to use the digital signature proposed by Camenisch and Lysyanskaya [9].

It is possible to use Zero Knowledge Set Membership or accumulators to validate user information without revealing it. A possible scenario is to perform KYC operations. For example, it would be possible to validate that the country of residence of a user is one belonging to the European Union, without revealing which country. Similarly, it is possible to validate membership in whitelists or blacklists, which would be important in the context of Anti-Money Laundering (AML) solutions for example.

The holy grail for privacy on DLT systems is the construction of private smart contracts. Ethereum [61] allows to construct smart contracts over blockchain, which can be seen as generic applications running in a distributed way, therefore avoiding the necessity to have a centralized solution. In other words, a smart contract is a piece of code that will run by all participants in Ethereum network. However, since there is no mechanism to provide privacy to the system, we have that all the information in smart contracts is visible by every other party, what constitutes a huge issue in many scenarios. This problem could be solved by using zk-SNARKs [37], but it requires a trusted setup, and this problem is even worse in the case of smart contracts, because we need a new setup for each contract. Hawk [41] is an interesting proposal to implement private smart contracts, however it not only needs a new setup for each contract, but also requires a trusted manager, who can view the user private information. Bulletproofs is an interesting proposal regarding private smart contracts, since it avoids the trusted setup and offers a generic ZKP protocol which has small proofs.

Recent breakthroughs in cryptography permit us to construct new protocols and achieve *privacy on demand*. These new cryptographic algorithms can be ultimately considered as tools that can be reused in different problems. ZKRP is an important tool that is necessary in order to construct a *toolbox* to deal with more complex applications.

Finally, an important research topic is the construction of post-quantum zero knowledge proofs. Recently, Benoît Libert et al. [42] proposed a construction of ZKRP based on lattices. However, the proof size is 3.54 MB for secret whose size is 2^{1000}. Although the secret is huge, the size of the proof can't be made considerably smaller when the secrets is smaller. Hence, optimizing this construction would allow to reduce the gap existing between conventional schemes and quantum-resistant ones.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Adida B (2008) Helios: web-based open-audit voting. In: Proceedings of the 17th conference on security symposium, SS'08, USENIX Association, Berkeley, pp 335–348
2. Baldimtsi F, Camenisch J, Dubovitskaya M, Lysyanskaya A, Reyzin L, Samelin K, Yakoubov S (2017) Accumulators with applications to anonymity-preserving revocation. In: 2017 IEEE European symposium on security and privacy, EuroS&P 2017, Paris, 26–28 April 2017, pp 301–315
3. Barreto P, Naehrig M (2006) Pairing-friendly elliptic curves of prime order. In: Bart P, Stafford T (eds) Selected areas in cryptography. Springer, Berlin, pp 319–331
4. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on computer and communications security, CCS '93, ACM, NY, pp 62–73

5. Boneh D, Bünz B, Fisch B (2018) Batching techniques for accumulators with applications to iops and stateless blockchains. Cryptology ePrint Archive, Report 2018/1188. https://eprint.iacr.org/2018/1188
6. Boneh D, Lynn B, Shacham H (2001) Short signatures from the weil pairing. In: Proceedings of the 7th international conference on the theory and application of cryptology and information security: advances in cryptology, ASIACRYPT '01, Springer, Berlin, pp 514–532
7. Bünz B, Bootle J, Boneh D, Poelstra A, Wuille P, Maxwell G (May 2018) Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE symposium on security and privacy (SP), pp 315–334
8. Camenisch J, Lysyanskaya A (2002) Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung M (ed) Advances in cryptology–CRYPTO 2002. Springer, Berlin, pp 61–76
9. Camenisch J, Lysyanskaya A (2003) A signature scheme with efficient protocols. In: Cimato S, Persiano G, Galdi C (eds) Security in communication networks. Springer, Berlin, pp 268–289
10. Camenisch J, Stadler M (1997) Efficient group signature schemes for large groups. In: Kaliski BS (ed) Advances in cryptology—CRYPTO '97. Springer, Berlin, pp 410–424
11. Camenisch J, Chaabouni R, Shelat A (2008) Efficient protocols for set membership and range proofs. In: Pieprzyk J (ed) Advances in cryptology—ASIACRYPT 2008. Springer, Berlin, pp 234–252
12. Camenisch J, Kohlweiss M, Soriente C (2009) An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Public key cryptography—PKC 2009, 12th international conference on practice and theory in public key cryptography, Irvine, CA, March 18–20, 2009. Proceedings, pp 481–500
13. Canard S, Coisel I, Jambert A, Traoré J (2014) New results for the practical use of range proofs. In: Katsikas S, Agudo I (eds) Public key infrastructures, services and applications. Springer, Berlin, pp 47–64
14. Canetti R, Goldwasser S, Ishai Y, Krawczyk H, Shi E, Tromer E, Venkitasubramaniam M, Zohar A. Zero knowledge proof standardization workshop. https://zkproof.org/index.html
15. Certicom Research (2000) SEC 2: recommended elliptic curve domain parameters. In: Standards for efficient cryptography
16. Chaabouni R (2017) Enhancing privacy protection set membership, range proofs, and the extended access control. p 239
17. Chaabouni R, Lipmaa H, Shelat A (2010) Additive combinatorics and discrete logarithm based range protocols. In: Steinfeld R, Hawkes P (eds) Information security and privacy. Springer, Berlin, pp 336–351
18. Chaabouni R, Lipmaa H, Zhang B (2012) A non-interactive range proof with constant communication. In: Keromytis AD (ed) Financial cryptography and data security. Springer, Berlin, pp 179–199
19. Chan A, Frankel Y, Tsiounis Y (1998) Easy come—easy go divisible cash. In: Nyberg K (ed) Advances in cryptology—EUROCRYPT'98. Springer, Berlin
20. Chaum D (1985) Security without identification: transaction systems to make big brother obsolete. Commun ACM 28(10):1030–1044
21. Chen G, Chen S, Xiao Y, Zhang Y, Lin Z, Lai TH (2018) SgxPectre attacks: stealing intel secrets from SGX enclaves via speculative execution. In: arXiv, Cornnell University Library. https://arxiv.org/abs/1802.09085 (visited on 19/09/2018)
22. Couteau G, Peters T, Pointcheval D (2017) Removing the strong rsa assumption from arguments over the integers. In: Coron J-S, Nielsen JB (eds) Advances in cryptology—EUROCRYPT 2017. Springer International Publishing, Cham, pp 321–350
23. Dagher G, Bünz B, Bonneau J, Clark J, Boneh D (2015) Provisions: privacy-preserving proofs of solvency for bitcoin exchanges. In: Proceedings of the 22Nd ACM SIGSAC conference on computer and communications security, CCS '15, ACM, NY, pp 720–731
24. Damgård IB (1990) Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser S (ed) Advances in cryptology—CRYPTO' 88. Springer, New York, pp 328–335
25. Damgård I (1995) Practical and provably secure release of a secret and exchange of signatures. J Cryptol 8(4):201–222
26. Damgård I, Jurik M, Nielsen JB (2010) A generalization of paillier's public-key system with applications to electronic voting. Int J Inf Secur 9(6):371–385
27. Dan B, Xavier B (2004) Short signatures without random oracles. In: Cachin C, Camenisch JL (eds) Advances in cryptology—EUROCRYPT 2004. Springer, Berlin, pp 56–73
28. Dan B, Eu-Jin G, Kobbi N (2005) Evaluating 2-DNF formulas on ciphertexts. In: Kilian J (ed) Theory of cryptography. Springer, Berlin, pp 325–341
29. Fabrice B (2000) Efficient proofs that a committed number lies in an interval. In: Preneel B (ed) Advances in cryptology—EUROCRYPT 2000. Springer, Berlin, pp 431–444
30. Fiat A, Shamir A (1987) How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in cryptology—CRYPTO' 86. Springer, Berlin, pp 186–194
31. Fujisaki E, Okamoto T (1997) Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski BS (ed) Advances in cryptology—CRYPTO '97. Springer, Berlin, pp 16–30
32. Galbraith S, Paterson K, Smart N (2008) Pairings for cryptographers. Discrete Appl Math 156(16):3113–3121 Applications of algebra to cryptography
33. Goldreich O (2006) Foundations of cryptography, vol 1. Cambridge University Press, New York
34. Goldwasser S, Micali S, Rackoff C (1985) The knowledge complexity of interactive proof-systems. In: Proceedings of the seventeenth annual ACM symposium on theory of computing, STOC '85, ACM, New York, pp 291–304
35. Groth J (2011) Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In: Proceedings of the 17th international conference on the theory and application of cryptology and information security, ASIACRYPT'11, Springer, Berlin, pp 431–448
36. Groth J (2005) Non-interactive zero-knowledge arguments for voting. In: Ioannidis J, Keromytis A, Yung M (eds) Applied cryptography and network security. Berlin HeidelbergSpringer, Berlin, Heidelberg, pp 467–482
37. Groth J (2010) Short pairing-based non-interactive zero-knowledge arguments. In: Abe M (ed) Advances in cryptology—ASIACRYPT 2010. Springer, Berlin, pp 321–340
38. Koblitz N (1992) CM-curves with good cryptographic properties. In: Feigenbaum J (ed) Advances in cryptology—CRYPTO '91. Springer, Berlin, pp 279–287
39. Koens T (2018) Consensus by trusted hardware. https://www.linkedin.com/pulse/consensus-trusted-hardware-tommy-koens
40. Koens T, Ramaekers C, van Wijk C. Efficient zero-knowledge range proofs in ethereum. ING media. https://www.ingwb.com/media/2122048/zero-knowledge-range-proof-whitepaper.pdf
41. Kosba A, Miller A, Shi E, Wen Z, Papamanthou C (2016) Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE symposium on security and privacy (SP), pp 839–858
42. Libert B, Ling S, Nguyen K, Wang H (2018) Lattice-based zero-knowledge arguments for integer relations. In: Shacham H, Boldyreva A (eds) Advances in cryptology—CRYPTO 2018. Springer International Publishing, Cham, pp 700–732

43. Lipmaa H (2003) On diophantine complexity and statistical zero-knowledge arguments. In: Laih C-S (ed) Advances in cryptology—ASIACRYPT 2003. Springer, Berlin, pp 398–415
44. Lipmaa H, Asokan N, Niemi V (2003) Secure vickrey auctions without threshold trust. In: Blaze M (ed) Financial cryptography. Springer, Berlin, pp 87–101
45. Lysyanskaya A, Rivest RL, Sahai A, Wolf S (2000) Pseudonym systems. In: Heys H, Adams C (eds) Selected areas in cryptography. Springer, Berlin, pp 184–199
46. Maxwell Gregory (2016) Confidential transactions. https://people.xiph.org/~greg/confidential_values.txt
47. Micali S, Rabin M (2014) Cryptography miracles, secure auctions, matching problem verification. Commun ACM 57(2):85–93
48. Morais E, Koens T, van Wijk C. Zero knowledge set membership. ING media. https://www.ing.com/Newsroom/All-news/Blockchain-innovation-improves-data-privacy-for-clients.htm
49. Morais E, Rudgers P, van Wijk C, Koens T, Ramaekers C (2018) Zero knowledge range proof implementation. Github. https://github.com/ing-bank/zkrangeproof
50. Parkes D, Rabin M, Shieber S, Thorpe C (2006) Practical secrecy-preserving, verifiably correct and trustworthy auctions. In: Proceedings of the 8th international conference on electronic commerce: the new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet, ICEC '06, ACM, New York, pp 70–81
51. Pedersen T (1992) Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum J (ed) Advances in cryptology—CRYPTO '91. Springer, Berlin, pp 129–140
52. Poelstra A (2016) Mimblewimble. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf
53. Rabin MO, Shallit JO (1986) Randomized algorithms in number theory. Commun Pure Appl Math 39(S1):S239–S256
54. Rabin MO, Mansour Y, Muthukrishnan S, Yung M (2012) Strictly-black-box zero-knowledge and efficient validation of financial transactions. In: Czumaj A, Mehlhorn K, Pitts A, Wattenhofer R (eds) Automata, languages, and programming. ICALP 2012. Lecture notes in computer science, vol 7391. Springer, Berlin, pp 738–749
55. Schnorr C-P (1991) Efficient signature generation by smart cards. J Cryptol 4(3):161–174
56. Schoenmakers B (2001) Some efficient zero-knowledge proof techniques. In: Workshop on cryptographic protocols
57. Schoenmakers B (2005) Interval proofs revisited. Slides presented at the international workshop on frontiers in electronic elections
58. Schwarz M, Weiser S, Gruss D, Maurice C, Mangard S (2017) Malware guard extension: using SGX to conceal cache attacks. In: Polychronakis M, Meier M (eds) Detection of intrusions and malware, and vulnerability assessment. Springer International Publishing, Cham, pp 3–24
59. The World Bank. Guidelines procurement under IBRD loans and IDA credits. The International Bank for Reconstruction and Development. http://siteresources.worldbank.org/INTPROCUREMENT/Resources/Procurement-Guidelines-November-2003.pdf
60. Uzunkol O, Kiraz MS (2018) Still wrong use of pairings in cryptography. Appl Math Comput 333:467–479
61. Wood G (2014) Ethereum: A secure decentralized transaction ledger. http://gavwood.com/paper.pdf