SCRAPE: Scalable Randomness Attested by Public Entities

Ignacio Cascudo[1]* and Bernardo David[2][3]**

[1] Aalborg University, Denmark
[2] Aarhus University, Denmark
[3] IOHK, Hong Kong

Abstract. Uniform randomness beacons whose output can be publicly attested to be unbiased are required in several cryptographic protocols. A common approach to building such beacons is having a number parties run a coin tossing protocol with guaranteed output delivery (so that adversaries cannot simply keep honest parties from obtaining randomness, consequently halting protocols that rely on it). However, current constructions face serious scalability issues due to high computational and communication overheads. We present a coin tossing protocol for an honest majority that allows for any entity to verify that an output was honestly generated by observing publicly available information (even after the execution is complete), while achieving both guaranteed output delivery and scalability. The main building block of our construction is the first Publicly Verifiable Secret Sharing scheme for threshold access structures that requires only $O(n)$ exponentiations. Previous schemes required $O(nt)$ exponentiations (where t is the threshold) from each of the parties involved, making them unfit for scalable distributed randomness generation, which requires $t = n/2$ and thus $O(n^2)$ exponentiations.

1 Introduction

The problem of obtaining a reliable source of randomness has been studied since the early days of cryptography. Whereas individual parties can choose to trust locally available randomness sources, it has been shown that local randomness sources can be subverted [BLN16,DPSW16] and many applications require a common public randomness source that is guaranteed to be unbiased by a potential adversary. This necessity inspired the seminal work on Coin Tossing by Blum [Blu81], which allows two or more parties to generate an output that is guaranteed to be uniformly random as long as at least one of the parties is honest (and given that the protocol terminates).

* Ignacio Cascudo acknowledges support from the Danish Council for Independent Research, grant no. DFF-4002-00367.

** This project has received funding from the European research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme (grant agreement No 669255).

The concept of a public *randomness beacon* that periodically issues fresh unpredictable and unbiased random values was proposed by Rabin [Rab83] in the context of contract signing and has found several other applications such as voting protocols [Adi08], generating public parameters for cryptographic schemes [BDF$^+$15,LW15], privacy preserving instant messaging [WCGFJ12,vdHLZZ15], and anonymous browsing [DMS04,GRFJ14]. More recently, blockchain [Nak08,GKL15] applications such as smart contracts [KMS$^+$16,B$^+$14], sharding [CDE$^+$16] and Proof-of-Stake based consensus protocols [KKR$^+$16] have increased the need for randomness sources [BCG15].

Rabin's concept of randomness beacons fits the above applications very nicely but the proposed implementation in [Rab83] relies on a trusted third party. The goal of this paper is to construct a distributed randomness beacon guaranteeing output delivery and uniformly distributed randomness for the parties that use the beacon as long as a majority of these parties are honest. Moreover, in many of the aforementioned applications, parties that do not necessarily participate in randomness generation but wish to audit the protocol execution must be able to attest *a posteriori* that the randomness source is reliable and unbiased. Hence, we aim at constructing a publicly verifiable randomness beacon and not only a protocol that outputs randomness to the parties actively involved in its execution.

1.1 Related Works

A natural solution for obtaining randomness beacons consists in using a coin tossing protocol as proposed by Blum [Blu81] with its messages posted to a public bulletin board for later verification (or broadcast among the parties). However, it is known that in case half or more of the parties are corrupted, the adversary can bias the output of the protocol or even prevent the honest parties from obtaining any output at all by aborting protocol execution at a given point [Cle86]. Assuming that a majority of the players are honest, it is possible to guarantee output delivery [RBO89] through threshold verifiable secret sharing (VSS) [CGMA85] given that a broadcast channel is available. Basically, given that a majority of n parties are honest, each party can secret share its input into n shares such that $n/2$ are enough to reconstruct the secret, sending one share to each involved party before starting the coin tossing protocol. While the adversary cannot recover any input (since it has at most $n/2 - 1$ shares of each input), the honest parties collective know at least $n/2$ shares, which they can use to reconstruct the inputs of parties who abort and then finish the protocol.

While a coin tossing protocol with guaranteed output delivery (G.O.D.) with a honest majority based on VSS provides a reliable source of randomness, this approach still has two main issues: 1. most VSS schemes require interaction between the dealer and the other parties, which hinders scalability and 2. only parties who actively participate in the protocol can verify that it was executed correctly. While non-interactive VSS [Fel87] solves the interaction problem, it does not allow the protocol execution to be independently verified by entities that did not actively participate. A natural way to allow for any entity to verify that the outputs produced by such protocols is indeed honestly generated is to substitute traditional VSS by publicly verifiable secret sharing (PVSS) schemes [Sta96], which allow for anybody to verify the validity of shares and reconstructed secrets through information that can be made publicly available without requiring direct interaction between any of the parties. Variations of this approach have been proposed in [KKR$^+$16,SJK$^+$16].

While [KKR$^+$16] instantiates a plain [RBO89]-style G.O.D. coin tossing protocol requiring communication among all parties (through a public ledger), [SJK$^+$16] reduces the communication complexity by partitioning parties into committees that internally run a protocol with publicly verifiable outputs. Later on, a *client* that only communicates to the *leader* of each committee (instead of talking to all parties) can aggregate these outputs to obtain publicly verifiable randomness. However, while the vanilla approach of [KKR$^+$16] achieves security assuming only an honest majority (meaning that the adversary corrupts less than half of all parties), the communication efficient approach of [SJK$^+$16] only achieves security against an adversary that corrupts less than a third of all parties. Moreover, provided that there is an honest majority, the protocol of [KKR$^+$16] guarantees that all parties get output regardless of which parties are corrupted, while in the protocols of [SJK$^+$16], even if only the client is corrupted, it can abort and prevent all other parties from receiving randomness.

Even though coin tossing with G.O.D. built through PVSS can potentially achieve scalability and public verifiability, current PVSS constructions [Sta96,FO98,Sch99,BT99,RV05,HV09,Jha11,JVSN14] suffer from high computational overhead. In general, the parties are required to each compute $O(nt)$ exponentiations to verify n shares of a secret with threshold t, which translates into $O(n^2)$ exponentiations since $t = n/2$ in our randomness beacon application [4]. This computational overhead arises because the main idea behind these schemes is to commit to the coefficients of a polynomial used for a Shamir Secret Sharing [Sha79] and encrypt the shares, later using the commitments to the coefficients to independently compute commitments to the shares, which are proven in zero-knowledge to correspond to the encrypted shares. This approach was originally put forth in [Sch99], which uses the Fiat-Shamir heuristic (and consequently the random oracle model) to obtain the necessary non-interactive zero-knowledge proofs. Later on, variations of this protocol in the plain model were proposed in [RV05,JVSN14], which substitute the zero-knowledge proofs by checks based on Paillier Encryption [Pai99], and in [HV09,Jha11], which propose a pairing based method for checking share validity.

Other approaches for constructing public randomness beacons have been considered in [BCG15,BDF$^+$15,LW15,BLMR14,BGM16]. Public verifiability (or au-

[4] In fact [Jha11] provides an alternative solution where only $O(n)$ exponentiations and a constant number of pairings are required for verification but $O(n)$ pairings are required for setup and $O(n^2)$ exponentiations in the target group of a bilinear map (more expensive than the other exponentiations performed in the source groups) are required for reconstruction.

ditability) in the context of general multiparty computation protocol has been previously considered in [BDO14,SV15].

1.2 Our Contributions

We introduce SCRAPE, a protocol that implements a publicly verifiable randomness beacon given an honest majority through a PVSS based guaranteed output delivery coin tossing protocol. Our main result lies at the core of SCRAPE: the *first* threshold PVSS scheme that only requires a *linear* number of exponentiations for sharing, verifying and reconstruction, whereas previous schemes only achieve quadratic complexity. This PVSS scheme can be instantiated both under the Decisional Diffie Hellman (DDH) assumption in the Random Oracle Model (ROM) and in the *plain model* under the Decisional Bilinear Square (DBS) assumption [HV09]. While improving on the computational complexity of previous schemes, our PVSS scheme retains a similarly low communication overhead, making it fit for applications with large amounts of users. We remark that our new PVSS schemes can also be used to improve the performance of [SJK$^+$16].

Model: As in previous works [BDO14], we assume that the parties can use a "public bulletin board" to publish information that will be used for posterior verification. In fact, in the applications we are interested in, a *ledger* where messages can be posted for posterior verification is readily available, since the Bitcoin Backbone protocol itself implements such a mechanism (*i.e.* the *distributed ledger* analysed in [GKL15]). Nevertheless, our protocols are compatible with any public ledger, not only with that of [GKL15]. Scalability is also a concern since hundreds of thousands of users are usually involved in randomness generation, making it simpler to post messages in a publicly accessible ledger rather than requiring all parties to communicate among themselves.

Our Techniques: We improve on Schoenmakers' PVSS scheme [Sch99] and its variants (which require $O(nt)$ exponentiations to verify n shares) by designing a share verification procedure that only requires $O(n)$ exponentiations (or pairings). Our procedure explores the fact that sharing a secret with Shamir Secret Sharing [Sha79] is equivalent to encoding the secret (plus randomness) with a Reed Solomon error correcting code, a fact which was first observed by McEliece *et al.* in [MS81]. Since shares from Shamir Secret Sharing form a codeword of a Reed Solomon code, computing the inner product of a share vector with a codeword from the corresponding dual code should yield 0 if the shares are correctly computed. As in [Sch99], the dealer in our scheme shares the secret using Shamir Secret sharing, encrypts the shares $s_1, \ldots, s_n$ in ciphertexts of the form $h^{sk_i s_i}$ (where h^{sk_i} is a public key and sk_i is a secret key) but also commits to all shares by computing $v_i = g^{s_i}$, where g, h are two independently chosen generators of a group where the DLOG problem is assumed to be hard. The dealer also provides evidence that the shares in the ciphertexts are the same as the shares in the commitments. To verify the validity of the shares, anybody can sample a random codeword $\boldsymbol{c}^{\perp} = (\boldsymbol{c}_1^{\perp}, \ldots, \boldsymbol{c}_n^{\perp})$ of the dual code of the Reed Solomon code corresponding to the instance of Shamir Secret sharing that was used, compute the inner product of $\boldsymbol{c}^{\perp}$ with the share vectors in the exponents of g (by computing $\prod_i v_i^{\boldsymbol{c}_i^{\perp}} = g^{\sum_i s_i \boldsymbol{c}_i^{\perp}}$) and check that it is equal to $g^0 = 1$. If the shares are not valid, this check fails with large probability. To prove that the shares in the ciphertexts and in the commitments are the same, the dealer can either use a non-interactive zero-knowledge (NIZK) proof constructed using the Fiat-Shamir heuristic as in [Sch99] (resulting in a construction in the ROM under the DDH assumption) or have the parties do pairing based checks as in [HV09] (resulting in a construction in the plain model under the DBS assumption).

Concrete Efficiency: In the DDH based construction in the ROM, the dealer is required to compute $4n$ exponentiations in the sharing phase, while verification and reconstruction respectively require $4n$ and $5t + 3$ exponentiations (given that all n shares are verified but only t shares are used in reconstruction). In the DBS based construction in the plain model, the dealer is required to compute $2n$ exponentiations in the sharing phase, while verification requires $2n$ pairings and reconstruction requires $2n$ pairings and $t + 1$ exponentiations (given that n decrypted shares are verified but only t shares are used in reconstruction). Previous results [Sch99,HV09] required roughly nt extra exponentiations in the verification phase, resulting in $n^2/2$ extra exponentiations in the randomness beacon application, which requires $t = n/2$. In the random oracle model construction, extra NIZK data is needed, amounting to a total of $2n$ group elements and $n+1$ ring elements published by the dealer. In the construction in the plain model, the dealer saves on the NIZK data and only posts $2n$ group elements, while requiring more expensive computation (*i.e.* pairings).

1.3 Organization

In Section 2 we introduce notation and definitions that will be used throughout the paper. In Section 3, we present our PVSS protocol based on the DDH assumption in the ROM. In Section 4, we introduce our PVSS protocol in the plain mode based on the DBS assumption. In Section 5, we construct a random beacon based on our PVSS protocols. In Section 6, we analyse the concrete complexity and performance of our protocols and present benchmarks based on a prototype implementation. Finally, in Section 7, we conclude with directions for future work.

2 Preliminaries

In this section, we establish notation and introduce definitions that will be used throughout the paper. We denote uniformly sampling a random element x from a finite set $\mathcal{D}$ by $x \leftarrow \mathcal{D}$. We denote vectors as $\boldsymbol{x} = (x_1, \ldots, x_n)$. We denote the inner product of two vectors $\boldsymbol{x}, \boldsymbol{y}$ as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{1 \leq i \leq n} x_i \cdot y_i$. For the sake of notation, the integer n will always be considered to be even, so that $n/2$ is an integer. In this paper q will always denote a prime number. We denote by $\mathbb{Z}_q$ the ring of integers modulo q and by $\mathbb{G}$ a finite multiplicative group of order q. Since q is prime, $\mathbb{Z}_q$ is a finite field and $\mathbb{G}$ is a cyclic group where every element $g \neq 1$ is a generator. We denote by $\mathbb{Z}_q[x]$ the ring of polynomials in one variable with coefficients in $\mathbb{Z}_q$. We denote by $log_g e$ the discrete logarithm of an element $e \in \mathbb{G}$ with respect to generator $g \in \mathbb{G}$.

2.1 Coding Theory

We define a $[n, k, d]$ code C to be a linear error correcting code over $\mathbb{Z}_q$ of length n, dimension k and minimum distance d. Its dual code $C^{\perp}$ is the vector space which consists of all vectors $\boldsymbol{c} \in \mathbb{Z}_q^n$ such that $\langle \boldsymbol{c}, \boldsymbol{c}^{\perp} \rangle = 0$ for all $\boldsymbol{c}$ in C. The dual code $C^{\perp}$ of an $[n, k, d]$ code C is an $[n, n-k, d^{\perp}]$ code (for some $d^{\perp}$). In this work, we will use the following basic linear algebra fact.

Lemma 1. *If $\boldsymbol{v} \in \mathbb{Z}_q^n \setminus C$, and $\boldsymbol{c}^{\perp}$ is chosen uniformly at random in $C^{\perp}$ then the probability that $\langle \boldsymbol{v}, \boldsymbol{c}^{\perp} \rangle = 0$ is exactly $1/q$.*

Proof. By linearity, a $\boldsymbol{c}^{\perp} \in C^{\perp}$ is orthogonal to $\boldsymbol{v}$ if only if it is also orthogonal to every vector in the code D spanned by $\boldsymbol{v}$ and C, i.e., if and only if $\boldsymbol{c}^{\perp} \in D^{\perp}$. Since $\boldsymbol{v} \notin C$, then the dimension of D is $k+1$ and hence the space $D^{\perp}$ has dimension $n-k-1$. Therefore if $\boldsymbol{c}^{\perp}$ is chosen uniformly at random in $C^{\perp}$ the probability that $\langle \boldsymbol{v}, \boldsymbol{c}^{\perp} \rangle = 0$ is

$$\frac{\#(D^{\perp})}{\#(C^{\perp})} = \frac{q^{n-k-1}}{q^{n-k}} = \frac{1}{q}.$$

Moreover, in this work we will always be under the assumption $n < q$ and we will use Reed-Solomon codes C of the following form

$$C = \{(p(1), p(2), \ldots, p(n)) : p(x) \in \mathbb{Z}_q[x], \deg p(x) \leq k-1\}$$

where $p(x)$ ranges over all polynomials in $\mathbb{Z}_q[x]$ of degree at most $k-1$. This is an $[n, k, n-k+1]$-code. Its dual $C^{\perp}$ is an $[n, n-k, k+1]$-code, which can be defined as follows

$$C^{\perp} = \{(v_1 f(1), v_2 f(2), \ldots, v_n f(n)) : f(x) \in \mathbb{Z}_q[x], \deg f(x) \leq n-k-1\}$$

for the coefficients $v_i = \prod_{j=1, j \neq i}^{n} \frac{1}{i-j}$.

2.2 Shamir Secret Sharing

An (n, t) threshold secret sharing scheme allows a *dealer* D to split a secret s into n shares $\boldsymbol{S} = (s_1, \ldots, s_n)$ distributed among n parties $P_1, \ldots, P_n$ such that it is possible to reconstruct the secret given t of the shares but no information at all is revealed if less shares are known. We refer to $\boldsymbol{S}$ as the *share vector* of the secret sharing scheme. The first threshold secret sharing scheme was introduced by Shamir in [Sha79]. In order to split a secret $s \in \mathbb{Z}_q$, the dealer samples $t-1$ random coefficients $c_1, \dots, c_{t-1} \leftarrow \mathbb{Z}_q$ and constructs a polynomial $p(x) = s + c_1 x + c_2 x^2 + \dots + c_{t-1} x^{t-1}$. The shares are computed as $s_i = p(i)$ for $1 \leq i \leq n$. A party who possesses t shares can use Lagrange interpolation to recover the polynomial $p(x)$ and thus obtain s. On the other hand, a party who knows less than t shares has no information about the secret. McEliece *et al.* first observed that sharing a secret into n shares with Shamir Secret Sharing is equivalent to encoding the message $(x, c_1, \dots, c_{t-1})$ under a $[n, t, n-t+1]$ Reed Solomon code, implying that the share vector $\boldsymbol{S}$ is a codeword of such Reed Solomon code.

2.3 Assumptions

One of our constructions is proven in the Random Oracle Model [BR93], where it is assumed that the parties are given access to a function $H(x)$ that takes inputs of any size and returns unique uniformly random outputs of fixed size (returning the same output every time the input is the same). Such a function can be instantiated in practice by a cryptographic hash function. In this model we prove security of our protocols under the DDH assumption, that states given g, g^α, g^β it is hard for a PPT adversary to distinguish between $g^{\alpha\beta}$ from g^r, where g is a generator of a group $\mathbb{G}$ of order q and $\alpha, \beta, r \leftarrow \mathbb{Z}_q$.

Bilinear Groups and the Decisional Bilinear Square assumption As in previous works [HV09] we have chosen to present our scheme over a symmetric bilinear group (*e.g.* Type-I in the terminology of [GPS06]). However, we remark that our construction can also be easily converted to asymmetric bilinear groups (*e.g.* Type-III in the terminology of [GPS06]) [AHO16], for which state-of-the-art pairing friendly curves [BN06] for which more efficient algorithms for computing pairings [AKL+11] are known.

Definition 1. ***Bilinear Group*** *A bilinear group is described by a tuple $\Lambda := (q, \mathbb{G}, \mathbb{G}_T, e)$ where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order q and e is a bilinear map $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:*

- ***Bilinearity*** *$e(g^\alpha, g^\beta) = e(g, g)^{\alpha\beta}$ for every $g \in \mathbb{G}$ and $\alpha, \beta \in \mathbb{Z}_q$.*
- ***Non-degeneration*** *$e(g, g) \neq 1$ unless $g = 1$.*
- ***Efficiency*** *There exist efficient algorithms to compute group operations in $\mathbb{G}, \mathbb{G}_T$ and to evaluate $e(x, y)$ for $x, y \in \mathbb{G}$.*

We prove our pairings based protocol secure under the Decisional Bilinear Square (DBS) assumption [HV09] that was shown in that paper to be equivalent to the Decisional Bilinear Quotient assumption [LV08] and related to the Decisional Bilinear Diffie Hellman assumption.

Assumption 1 ***Decisional Bilinear Square (DBS) [HV09]*** *Let $\Lambda := (q, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear group. For a generator $g \in \mathbb{G}$, random values $\mu, \nu, s \leftarrow \mathbb{Z}_q$ and given $u = g^\mu$ and $v = g^\nu$, the following probability distributions are computationally indistinguishable: $D_0 = (g, u, v, T_0 = e(u, u)^\nu)$ and $D_1 = (g, u, v, T_1 = e(u, u)^s)$.*

Adversarial Model We prove the security of our protocols in the stand alone setting against malicious adversaries, who may deviate from the protocol in any arbitrary way. We consider static adversaries, who have to choose which parties to corrupt before protocol execution begins.

Public Ledger and Broadcast Channel We assume that the parties have access to a public ledger with *Liveness*, meaning that an adversary cannot prevent honest parties from adding information and agreeing on it, and *Persistence*, meaning that the information cannot be modified or removed a posteriori. It is known that such a ledger can be implemented by the Bitcoin backbone protocol assuming an honest majority, digital signatures and a Random Oracle [GKL15]. However, we remark that our protocols do not rely on any properties that are unique to the ledger of [GKL15], meaning that our constructions can also be instantiated over public ledgers in the plain model. Notice that access to a broadcast protocol is commonly assumed in multiparty protocols for an honest majority [RBO89] and that the same effect of broadcasting a messages can be achieved by writing it to the ledger. We also remark that the availability of a *public bulletin board* has been assumed in previous works on public verifiability for multiparty protocols [BDO14,SV15].

2.4 Publicly Verifiable Secret Sharing

We adopt the general model for PVSS schemes of [Sch99] and the security definitions of [RV05,HV09] (with some differences that we remark below). We consider a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and a dealer D who shares a secret among all the parties in $\mathcal{P}$. We will construct schemes for (n, t)-threshold access structures, meaning that the secret is split in n shares in such a way that knowing at most $t - 1$ shares reveals no information but a collection of t shares allows for secret reconstruction. Additionally, any external verifier V can check that the D is acting honest without learning any information about the shares or the secret. A PVSS protocol has four phases described below:

- **Setup** The dealer D generates and publishes the parameters of the scheme. Every party P_i publishes a public key pk_i and withholds the corresponding secret key sk_i.
- **Distribution** The dealer creates shares $s_1, \ldots, s_n$ for the secret s, encrypts share s_i with the public key pk_i for $i = 1, \ldots, n$ and publishes these encryptions $\hat{s}_i$, together with a proof $PROOF_D$ that these are indeed encryptions of a valid sharing of some secret.
- **Verification** In this phase, any external V (not necessarily being a participant in the protocol) can verify non-interactively, given all the public information until this point, that the values $\hat{s}_i$ are encryptions of a valid sharing of some secret.
- **Reconstruction** This phase is divided in two.
  Decryption of the shares: This phase can be carried out by any set $\mathcal{Q}$ of t or more parties. Every party P_i in $\mathcal{Q}$ decrypts the share s_i from the ciphertext $\hat{s}_i$

by using its secret key sk_i, and publishes s_i together with a (non-interactive) zero-knowledge proof $PROOF_i$ that this value is indeed a correct decryption of $\hat{s}_i$.

Share pooling: Any external verifier V (not necessarily being a participant in the protocol) can now execute this phase. V first checks whether the proofs $PROOF_i$ are correct. If the check passes for less than t parties in $\mathcal{Q}$ then V aborts; otherwise V applies a reconstruction procedure to the set s_i of shares corresponding to parties P_i that passed the checks.

A PVSS scheme must provide three security guarantees: Correctness, Verifiability and IND1-Secrecy. These properties are defined below:

– **Correctness** If the dealer and all players in $\mathcal{Q}$ are honest, then all verification checks in the verification and reconstruction phases pass and the secret can be reconstructed from the information published by the players in $\mathcal{Q}$ in the reconstruction phase.
– **Verifiability** If the check in the verification step passes, then with high probability the values $\hat{s}_i$ are encryptions of a valid sharing of some secret. Furthermore if the check in the Reconstruction phase passes then the communicated values s_i are indeed the shares of the secret distributed by the dealer.
– **IND1-Secrecy** Prior to the reconstruction phase, the public information together with the secret keys sk_i of any set of at most $t-1$ players gives no information about the secret. Formally this is stated as in the following indistinguishability based definition adapted from [RV05,HV09]:

Definition 2. ***Indistinguishability of secrets (IND1-secrecy)*** *We say that the PVSS is IND1-secret if for any polynomial time adversary $\mathcal{A}_{Priv}$ corrupting at most $t-1$ parties, $\mathcal{A}_{Priv}$ has negligible advantage in the following game played against a challenger.*

1. *The challenger runs the Setup phase of the PVSS as the dealer and sends all public information to $\mathcal{A}_{Priv}$. Moreover, it creates secret and public keys for all uncorrupted parties, and sends the corresponding public keys to $\mathcal{A}_{Priv}$.*
2. *$\mathcal{A}_{Priv}$ creates secret keys for the corrupted parties and sends the corresponding public keys to the challenger.*
3. *The challenger chooses values x_0 and x_1 at random in the space of secrets. Furthermore it chooses $b \leftarrow \{0,1\}$ uniformly at random. It runs the Distribution phase of the protocol with x_0 as secret. It sends $\mathcal{A}_{Priv}$ all public information generated in that phase, together with x_b.*
4. *$\mathcal{A}_{Priv}$ outputs a guess $b' \in \{0,1\}$.*

The advantage of $\mathcal{A}_{Priv}$ is defined as $|\Pr[b = b'] - 1/2|$.

The IND1-secrecy definition is the one used in [RV05,HV09], except for the fact that we do not impose any privacy requirement after the Reconstruction phase. The difference stems from the fact that in [RV05,HV09] it was required that nobody learns the secret but the parties interacting during the reconstruction, while in our random beacon application the secret must be publicly reconstructed and published. We remark that our scheme can achieve both the relaxed definition required by the random beacon application and the stronger secrecy guarantees of [RV05,HV09] (through the use of private channels between parties or through the technique of [HV09] that requires extra data to be posted to the ledger). We also remark that our schemes can achieve the stronger secrecy notion formalized as IND2-secrecy in [RV05,HV09], which allows the adversary to choose arbitrary secrets. This is done by a black box transformation to the protocols that allows for sharing arbitrary secrets instead of random ones by using the random shared secret as a "one time pad" to encrypt an arbitrary secret, which is formally proven in [RV05,HV09].

2.5 Commitments

Commitment schemes [Blu81] are a fundamental cryptographic primitive that function as a digital safe deposit box. Basically, a sender commits to a message m by putting it inside the box, locking the box and sending the box to a receiver. Later on, the sender can open the commitment by giving the receiver the key to the box, revealing m. Notice that the sender cannot change the message after it gives the locked box to the receiver (a property called *binding*) while the receiver cannot learn the message before he receives the key (a property called *hiding*). For formal definitions and constructions of commitment schemes with various security guarantees and very good efficiency we refer the readers to the following works for the stand alone [Nao91] and the Universal Composability [CDD$^+$16] models. We define a general syntax for commitments as follows:

- $\mathsf{Com}(m, r)$ takes as input a message m and randomness r, outputting a commitment Com to message m.
- $\mathsf{Open}(m, r)$ takes as input a message m and randomness r, outputting the opening information necessary for checking whether a corresponding commitment Com is valid with respect to m and r.

2.6 Zero-Knowledge Proofs of Discrete Logarithm Knowledge

In our construction based on the DDH assumption in the random oracle model we will need a zero-knowledge proof of knowledge of a value $\alpha \in \mathbb{Z}_q$ such that $x = g^\alpha$ and $y = h^\alpha$ given g, x, h, y. We denote this proof by $DLEQ(g, x, h, y)$. Chaum and Pedersen constructed a sigma protocol to perform this proof in [CP93], their protocol works as follows:

1. The prover computes $a_1 = g^w$ and $a_2 = h^w$ where $w \leftarrow \mathbb{Z}_q$ and sends a_1, a_2 to the verifier.
2. The verifier sends a challenge $e \leftarrow \mathbb{Z}_q$ to the prover.
3. The prover sends a response $z = w - \alpha e$ to the verifier.

4. The verifier checks that $a_1 = g^z x^e$ and $a_2 = h^z y^e$ and accepts the proof if this holds.

This proof has the properties of completeness, soundness and zero-knowledge. In our proofs, we will specifically reference the soundness property, which means that a prover cannot convince a verifier of a fake statement except with a negligible *soundness error* ϵ. Notice that this sigma protocol can be transformed into a non-interactive zero-knowledge proof of knowledge of α in the random oracle model through the Fiat-Shamir heuristic [FS87,PS96]. We remark that, as in [Sch99], we need to compute this proof in parallel for n distinct pairs of values $(x_1, y_1), \dots, (x_n, y_n)$. In this case, a single challenge e is computed by the prover as $e = H(x_1, y_1, \dots, x_n, y_n, a_{1,1}, a_{2,1}, \dots, a_{1,n}, a_{2,n})$, where the values $a_{1,i}, a_{2,i}$ are computed according to x_i, y_i as described above and $H(\cdot)$ is a random oracle (that can be of course substituted by a cryptographic hash function). The proof then consists of the challenge e along with responses z_i computed according to each x_i, y_i. The verifier can check the proof by computing $a'_{1,i} = g^{z_i} x_i^e$ and $a'_{2,i} = h^{z_i} y_i^e$, and verifying that $H(x_1, y_1, \dots, x_n, y_n, a'_{1,1}, a'_{2,i}, \dots, a'_{1,n}, a'_{2,n}) = e$.

3 PVSS based on the DDH assumption in the ROM

In this section, we construct a PVSS protocol secure under the DDH assumption in the Random Oracle Model. Our general approach is similar to that of Schoenmakers [Sch99] but differs significantly in the procedure used for share verification, which represents the main overhead in Schoenmakers' scheme. In a setup phase, each party P_i is required to register in the ledger (or broadcast) public keys pk_i of the form $pk_i = h^{sk_i}$, where h is a generator of group $\mathbb{G}_q$ of order p and $sk_i \leftarrow \mathbb{Z}_q$ is a secret key stored by each party. In the Distribution phase, the dealer starts by sharing a secret $s \leftarrow \mathbb{Z}_q$ with Shamir Secret Sharing and encrypting the shares $s_1, \dots, s_n$ under the parties registered public keys by computing $\hat{s}_i = pk_i^{s_i}$, aiming at sharing a random secret of the form h^s. However, instead of committing to the coefficients of the polynomial used for Shamir Secret Sharing, we commit to all the shares using an independently chosen generator g of $\mathbb{G}_q$ by publishing $v_i = g^{s_i}$. Moreover, the dealer publishes non-interactive zero knowledge proofs that the ciphertexts $\hat{s}_i$ contain the same shares as the commitments v_i (using DLEQ as described in Section 2.6). Notice that since g is chosen independently from h, knowing g^s does not help an adversary retrieve the secret h^s unless it can compute the discrete log $log_h g$. In the Verification phase, anybody observing the public encrypted shares $\hat{s}_i$, commitments to shares v_i and NIZKs can check that the encrypted shares were correctly generated by first verifying that the shares in $\hat{s}_i$ indeed are the same in v_i and then performing an information theoretical check that only requires n exponentiations. This check consists in selecting a codeword $\boldsymbol{c}^\perp = (\boldsymbol{c}_1^\perp, \dots, \boldsymbol{c}_n^\perp)$ from the dual code $C^\perp$ corresponding to the Reed Solomon code C to which the Shamir Secret Sharing procedure used in Distribution is equivalent, computing the inner product of $\boldsymbol{c}^\perp$ with the shares vector $(s_1, \dots, s_n)$ by computing

Protocol π_{DDH}

Let g and h be two independently chosen generators of a group $\mathbb{G}_q$ of order q. Let $H(\cdot)$ be a random oracle. Let C be the linear error correcting code corresponding to the (n,t)-threshold Shamir Secret Sharing scheme and let $C^{\perp}$ be its dual code. Protocol π_{DDH} is run between n parties $P_1, \dots, P_n$, a dealer D and an external verifier V (in fact any number of external verifiers) who have access to a public ledger where they can post information for later verification. The protocol proceeds as follows:

1. **Setup:** Party P_i generates a secret key $sk_i \leftarrow \mathbb{Z}_q$, a public key $pk_i = h^{sk_i}$ and registers the public key pk_i by posting it to the public ledger, for $1 \le i \le n$.
2. **Distribution** The dealer D first samples $s \leftarrow \mathbb{Z}_q$. The secret is defined to be $S = h^s$. D chooses $t-1$ coefficients $c_1, \dots, c_{t-1} \leftarrow \mathbb{Z}_q$. D constructs a polynomial $p(x) = s + c_1 x + c_2 x^2 + \cdots + c_{t-1}x^{t-1}$ and computes shares $s_i = p(i)$ for $1 \le i \le n$. D encrypts the shares as $\hat{s}_i = pk_i^{s_i}$, computes commitments $v_i = g^{s_i}$ and computes $DLEQ(g, v_i, pk_i, \hat{s}_i)$, for $1 \le i \le n$ from a single challenge e as described in Section 2.6, obtaining $(e, z_1, \dots, z_n)$. D publishes in the public ledger the encrypted shares $(\hat{s}_1, \dots, \hat{s}_n)$ along with $PROOF_D = (v_1, \dots, v_n, e, z_1, \dots, z_n)$.
3. **Verification:** The verifier V first checks that the $DLEQ(g, v_i, pk_i, \hat{s}_i)$ provided by D is valid as described in Section 2.6. If the proof is valid, V samples a random codeword $\boldsymbol{c}^{\perp} = (\boldsymbol{c}_1^{\perp}, \dots, \boldsymbol{c}_n^{\perp})$ of the dual code $C^{\perp}$ corresponding to the instance of Shamir's (n,t)-threshold secret sharing used by D and considers the shares valid if and only if the following expression is true:
$$\prod_{i=1}^{n} v_i^{\boldsymbol{c}_i^{\perp}} = 1.$$
4. **Reconstruction:** If a set of t or more parties $\mathcal{Q}$ wishes to reconstruct the secret, each party $P_i \in \mathcal{Q}$ starts by publishing in the public ledger its decrypted share $\tilde{s}_i = \hat{s}_i^{\frac{1}{sk_i}} = h^{s_i}$ and $PROOF_i = DLEQ(h, pk_i, \tilde{s}_i, \hat{s}_i)$ (showing that the decrypted share $\tilde{s}_i$ corresponds to $\hat{s}_i$). Once every party in $\mathcal{Q}$ publishes their decrypted shares and $PROOF_i$, they first verify that the proofs are valid and, if this check succeeds, reconstruct the secret $S = h^s$ by Lagrange interpolation:
$$\prod_{P_i \in \mathcal{Q}} (\tilde{s}_i)^{\lambda_i} = \prod_{P_i \in \mathcal{Q}} h^{p(i)\lambda_i} = h^{p(0)} = h^s,$$
where $\lambda_i = \prod_{j \ne i} \frac{j}{j-i}$ are the Lagrange coefficients.

Fig. 1. Protocol π_{DDH}

$\prod_i v_i^{\boldsymbol{c}_i^{\perp}} = g^{\sum_i s_i \boldsymbol{c}_i^{\perp}}$ and checking the result is $g^0 = 1$. The Reconstruction phase proceeds as in [Sch99], with each party P_i "decrypting" its share to obtain h^{s_i}, which it published along with a proof that it corresponds to the encrypted share $\hat{s}_i$. Once t decrypted shares are available, the parties can check that they are valid and use Lagrange interpolation to reconstruct the secret h^s. The protocol is described in Figure 1.

3.1 Security Analysis

Notice that the Setup and Reconstruction phases are exactly equal to those of [Sch99], while our protocol differs in the Distribution and Verification phases, where we apply our new technique. The key observation is that maliciously generated encrypted shares $\hat{s}_1, \dots, \hat{s}_n$ will only pass the verification procedure with probability $1/q$ plus the soundness error of the DLEQ proof, while $v_1, \dots, v_n$ reveal no information about the secret h^s under the DDH assumption (by an argument similar to that of [Sch99]).

We formalize these observations below. First we consider IND1-secrecy. We remark that while we use our relaxed IND1-secrecy notion or our randomness beacon application (where no secrecy is preserved after reconstruction), Protocol π_{DDH} achieves the original stronger IND1-secrecy notion of [RV05,HV09] (where secrecy against parties outside the qualified set is guaranteed even after the reconstruction) if the reconstruction is carried out through private channels between the parties in the qualified set.

Theorem 1. *Under the decisional Diffie-Hellman assumption, the protocol π_{DDH} is IND1-secret against a static PPT adversary.*

Proof. We show that, if there exists an adversary $\mathcal{A}_{\text{Priv}}$ which can break the IND1-secrecy property of protocol π_{DDH}, then there exists an adversary $\mathcal{A}_{\text{DDH}}$ which can use $\mathcal{A}_{\text{Priv}}$ to break the decisional Diffie-Hellman assumption with the same advantage. Without loss of generality we assume $\mathcal{A}_{\text{Priv}}$ corrupts the $t-1$ first parties.

Let $(g, g^\alpha, g^\beta, g^\gamma)$ be an instance of the DDH problem. Obviously if $\alpha = 0$ or $\beta = 0$ then the problem is trivial, so we assume these values are nonzero. Now $\mathcal{A}_{\text{DDH}}$, using $\mathcal{A}_{\text{Priv}}$, can simulate an IND1 game as follows:

1. The challenger sets $h = g^\alpha$ and runs the Setup phase of π_{DDH}. For $t \leq i \leq n$, $\mathcal{A}_{\text{DDH}}$ selects uniformly random values $u_i \leftarrow \mathbb{Z}_p$ (these can be thought of implicitly defining sk_i as $sk_i = u_i/\alpha$) and sends the values $pk_i = g^{u_i}$ to $\mathcal{A}_{\text{Priv}}$.
2. For $1 \leq i \leq t-1$, $\mathcal{A}_{\text{Priv}}$ chooses uniformly random values $sk_i \leftarrow \mathbb{Z}_q$ and sets $pk_i = h^{sk_i}$ and sends this to the challenger.
3. For $1 \leq i \leq t-1$, the challenger chooses uniformly random values $s_i \leftarrow \mathbb{G}_q$ and sets $v_i = g^{s_i}$ and $\hat{s}_i = pk_i^{s_i}$.
   For $t \leq i \leq n$, it generates the values $v_i = g^{p(i)}$ where $p(x)$ is the unique polynomial of degree at most t determined by $p(0) = \beta$ and $p(i) = s_i$ for $i = 1, \dots, t-1$. Note that $\mathcal{A}_{\text{DDH}}$ does not know β, but it does know $g^\beta = g^{p(0)}$ and $g^{s_i} = g^{p(i)}$ for $1 \leq i \leq t-1$, so it can use Lagrange interpolation in the exponent to compute the adequate v_i. It also creates the values $\hat{s}_i = v_i^{u_i}$. Note that then $\hat{s}_i = g^{u_i \cdot p(i)} = pk_i^{p(i)}$. From all the computed values, the

challenger now creates the DLEQ proofs as the dealer does in the PVSS protocol. Finally it sends all this information together with the value g^{γ} (which plays the role of x_b in the IND game) to $\mathcal{A}_{\text{Priv}}$.

4. $\mathcal{A}_{\text{Priv}}$ makes a guess b'.

If $b' = 0$, $\mathcal{A}_{\text{DDH}}$ guesses that $\gamma = \alpha \cdot \beta$. If $b' = 1$, $\mathcal{A}_{\text{DDH}}$ guesses that γ is a random element in $\mathbb{Z}_p$.

The information that $\mathcal{A}_{\text{Priv}}$ receives in step 3. is distributed exactly like a sharing of the value $h^{\beta} = g^{\alpha \cdot \beta}$ with the PVSS. Consequently, $\gamma = \alpha \cdot \beta$ if and only if the value g^{γ} sent to $\mathcal{A}_{\text{Priv}}$ is the secret shared by the PVSS. It is now easy to see that the guessing advantage of $\mathcal{A}_{\text{DDH}}$ is the same as the advantage of $\mathcal{A}_{\text{Priv}}$.

The following two theorems guarantee the verifiability property of π_{DDH}.

Theorem 2. *If the dealer does not construct values $(v_i, \hat{s}_i)$ of the right form in the Distribution phase (i.e. either $log_g v_i \neq log_{pk_i} \hat{s}_i$ for some i, or $log_g v_i = log_{pk_i} \hat{s}_i = s_i$ for all i but the values s_i do not constitute a valid sharing of some secret in $\mathbb{Z}_q$ with the (n, t)-threshold Shamir secret sharing scheme), then this is detected in the verification step with probability at least $1 - \epsilon - 1/q$, where ϵ is the soundness error of the proof DLEQ.*

Proof. If the verification of DLEQ passes, then we have that, except with probability ϵ, for every $1 \leq i \leq n$, there exists s_i with $v_i = g^{s_i}$ and $\hat{s}_i = pk_i^{s_i}$. Now the values s_i are a valid sharing with the (n, t)-threshold Shamir secret sharing scheme if and only if the vector $\boldsymbol{v} = (s_1, \ldots, s_n) \in C$. Suppose that $(s_1, \ldots, s_n) \notin C$. Then by Lemma 1, since $\boldsymbol{c}^{\perp}$ is sampled uniformly at random then $\langle \boldsymbol{v}, \boldsymbol{c}^{\perp} \rangle \neq 0$ except with probability $1/q$. But then $\prod_{i=1}^{n} v_i^{\boldsymbol{c}_i^{\perp}} = \prod_{i=1}^{n} g^{s_i \cdot \boldsymbol{c}_i^{\perp}} = g^{\langle \boldsymbol{v}, \boldsymbol{c}^{\perp} \rangle} \neq 1$. Hence if the values s_i are not a valid Shamir sharing, then the check fails with probability $1 - 1/q$.

Theorem 3. *If a party in $\mathcal{Q}$ communicates an erroneous decryption share $\tilde{s}_i$ in the Reconstruction phase, then this is detected by the verifier with probability $1 - \epsilon$, where ϵ is the soundness error of the DLEQ proof.*

Proof. This is straightforward by definition since an adversary that succeeds in providing a DLEQ proof for an invalid decrypted share breaks DLEQ's soundness property.

4 PVSS based on pairings in the plain model

In this section, we construct a PVSS scheme based on the DBS assumption in the plain model (without requiring random oracles). This scheme uses the techniques of [HV09] to eliminate the need for the random oracle based NIZKs and instead use pairings to check that the encrypted shares $\hat{s}_i$ correspond to the committed shares v_i and, later on, check that the decrypted shares correspond to $\hat{s}_i$. We use the same information theoretical verification procedure as in the DDH based scheme. The protocol is described in Figure 2.

Protocol π_{DBS}

Let $\Lambda := (q, \mathbb{G}, \mathbb{G}_T, e)$ be a description of a bilinear group and g, h be two independently chosen generators of $\mathbb{G}$. Let C be the linear error correcting code equivalent to the (n, t)-threshold Shamir Secret Sharing scheme and $C^\perp$ its dual code. Protocol π_{DBS} is run between n parties $P_1, \ldots, P_n$, a dealer D and an external verifier V (in fact any number of external verifiers) who have access to a public ledger where they can post information for later verification. The protocol proceeds as follows:

1. **Setup:** Party P_i generates a secret key $sk_i \leftarrow \mathbb{Z}_q$, a public key $pk_i = h^{sk_i}$ and registers the public key pk_i by posting it to the public ledger, for $1 \le i \le n$.
2. **Distribution** The dealer D first samples $s \leftarrow \mathbb{Z}_q$. The secret is defined to be $S = e(h, h)^s$. D chooses $t - 1$ coefficients $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$. D constructs a polynomial $p(x) = s + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1}$ and computes shares $s_i = p(i)$ for $1 \le i \le n$. D encrypts the shares as $\hat{s}_i = pk_i^{s_i}$ and computes commitments $v_i = g^{s_i}$, for $1 \le i \le n$. D posts in the public ledger the encrypted shares $\hat{s}_1, \ldots, \hat{s}_n$ and $PROOF_D = (v_1, \ldots, v_n)$.
3. **Verification:** The verifier V first checks that $e(\hat{s}_i, g) = e(pk_i, v_i)$, for $1 \le i \le n$. If this check succeeds, V samples a random codeword $\boldsymbol{c}^\perp = (\boldsymbol{c}_1^\perp, \ldots, \boldsymbol{c}_n^\perp)$ of the dual code $C^\perp$ corresponding to the instance of Shamir's (n, t)-threshold secret sharing used by D and considers the shares valid if and only if the following expression is true:
$$\prod_{i=1}^{n} v_i^{\boldsymbol{c}_i^\perp} = 1.$$
4. **Reconstruction:** If a set of t or more parties $\mathcal{Q}$ wishes to reconstruct the secret, each party $P_i \in \mathcal{Q}$ starts publishing in the public ledger its decrypted share $\tilde{s}_i = \hat{s}_i^{\frac{1}{sk_i}} = h^{s_i}$ (here $PROOF_i$ is an empty string). Once every party in $\mathcal{Q}$ publishes their decrypted shares, they first verify that $e(pk_i, \tilde{s}_i) = e(\hat{s}_i, h)$ for every $P_i \in \mathcal{Q}$. If this check succeeds, they reconstruct the value h^s by Lagrange interpolation:
$$\prod_{P_i \in \mathcal{Q}} (\tilde{s}_i)^{\lambda_i} = \prod_{P_i \in \mathcal{Q}} h^{p(i)\lambda_i} = h^{p(0)} = h^s,$$
where $\lambda_i = \prod_{j \ne i} \frac{j}{j-i}$ are the Lagrange coefficients. The secret is then computed as $S = e(h^s, h)$.

Fig. 2. Protocol π_{DBS}

4.1 Security Analysis

As in the DDH based protocol, notice that the Setup and Reconstruction phases are exactly equal to those of [HV09], while our protocol differs in the Distribution and Verification phases. Maliciously generated encrypted shares $\hat{s}_1, \ldots, \hat{s}_n$ will only pass the verification procedure with probability of $1/q$, while $v_1, \ldots, v_n$ again reveal no information about the secret $e(h, h)^s$ but this time under the BDS assumption. Again we remark that Protocol π_{DBS} achieves the original stronger

IND1-secrecy notion of [RV05,HV09] (where secrecy against parties outside the qualified set is guaranteed even after the reconstruction) if the reconstruction is carried out through private channels between the parties in the qualified set.

Theorem 4. *Under the DBS assumption, protocol π_{DBS} is IND1-secret against a static PPT adversary.*

Proof. The proof is similar to that of Theorem 1. We want to show that, if there exists an adversary $\mathcal{A}_{\mathrm{Priv}}$ which can break the privacy property of protocol π_{DBS}, then there exists an adversary $\mathcal{A}_{\mathrm{BDS}}$ that breaks the decisional Diffie-Hellman assumption with the same advantage.

Suppose that we are given an instance $(g, g^\alpha, g^\beta, T)$ and the task of $\mathcal{A}_{\mathrm{BDS}}$ is to guess whether $T = e(g^\alpha, g^\alpha)^\beta$ or $T = e(g^\alpha, g^\alpha)^\gamma$ for a random $\gamma \leftarrow \mathbb{Z}_q$. $\mathcal{A}_{\mathrm{BDS}}$ now simulates an IND1 game for π_{DBS} between a challenger an $\mathcal{A}_{\mathrm{Priv}}$ following exactly the same steps as $\mathcal{A}_{\mathrm{DDH}}$ did for protocol π_{DDH} in the proof of Theorem 1, except the secret is now $e(h,h)^\beta = e(g^\alpha, g^\alpha)^\beta$ and at the end of step 3. the challenger sends the value T (rather than g^γ).

Now if the guess of $\mathcal{A}_{\mathrm{Priv}}$ is $b' = 0$, then $\mathcal{A}_{\mathrm{BDS}}$ guesses that $T = e(g^\alpha, g^\alpha)^\beta$. If $b' = 1$, it guesses that $T = e(g^\alpha, g^\alpha)^\gamma$ or a random $\gamma \leftarrow \mathbb{Z}_q$.

It is now easy to see that the advantage of $\mathcal{A}_{\mathrm{BDS}}$ is the same as that of $\mathcal{A}_{\mathrm{Priv}}$.

Theorem 5. *If the dealer does not construct values $(v_i, \hat{s}_i)$ of the right form in the Distribution phase (i.e. either $log_g v_i \neq log_{pk_i} \hat{s}_i$ for some i, or $log_g v_i = log_{pk_i} \hat{s}_i = s_i$ for all $1 \leq i \leq n$ but the values s_i do not constitute a valid sharing of some secret in $\mathbb{Z}_q$ with the (n,t)-threshold Shamir secret sharing scheme), then this is detected in the verification step with probability at least $1 - 1/q$.*

Proof. The only difference between this proof and the one for the protocol π_{DDH} is that here we do not use DLEQ proofs to guarantee that $log_{g_i} v_i = log_{pk_i} \hat{s}_i$ for all i. Instead this is verified by checking that $e(\hat{s}_i, g) = e(pk_i, v_i)$, for $1 \leq i \leq n$. Note that if $a = log_g v_i \neq log_{pk_i} \hat{s}_i = b$ for some i, then $e(\hat{s}_i, g) = e(pk_i, g)^b \neq e(pk_i, g)^a = e(pk_i, v_i)$ and the check fails with probability 1. The rest of the proof works exactly as in the case of π_{DDH}.

Theorem 6. *If a party in $\mathcal{Q}$ communicates an erroneous decryption share $\tilde{s}_i$ in the Reconstruction phase, then this is detected by the verifier with probability* 1

Proof. If $\tilde{s}_i = h^a$ with $a \neq s_i$ then $e(pk_i, \tilde{s}_i) = e(pk_i, h)^a \neq e(pk_i, h)^{s_i} = e(\hat{s}_i, h)$.

5 Building the SCRAPE Randomness Beacon

Publicly verifiable secret sharing schemes have a multitude of applications as discussed in [Sch99], among them universally verifiable elections, threshold versions of El Gamal encryption and threshold software key escrow. However, we are specially interested in constructing SCRAPE, a protocol that implements a distributed randomness beacon that is guaranteed to be secure given an honest

Protocol π_{SCRAPE}

Protocol π_{DDH} is run between n parties $P_1, \ldots, P_n$ who have access to a public ledger where they can post information for later verification. A PVSS protocol is used as a sub protocol and it is assumed that the Setup phase is already done and the public keys pk_i of each party P_i are already registered in the ledger.The protocol proceeds as follows:

1. **Commit:** For $1 \leq j \leq n$, party P_j executes the Distribution phase of the PVSS sub protocol as the Dealer with threshold $t = \frac{n}{2}$, publishing the encrypted shares $\hat{s}_1^j, \ldots, \hat{s}_n^j$ and the verification information $PROOF_D^j$ on the public ledger, and also learning the random secret h^{s^j} and s^j. P_j also publishes a commitment to the secret exponent $\mathsf{Com}(s^j, r_j)$ (with fresh randomness $r_j \leftarrow \mathbb{Z}_q$), for $1 \leq j \leq n$.
2. **Reveal:** For every set of encrypted shares $\hat{s}_1^j, \ldots, \hat{s}_n^j$ and the verification information $PROOF_D^j$ published in the public ledger, all parties run the Verification phase of the PVSS sub protocol. Let $\mathcal{C}$ be the set of all parties who published commitments and valid shares. Once $\frac{n}{2}$ parties have posted their commitments and valid shares on the ledger, party P_j opens its commitment, posting $\mathsf{Open}(s^j, r_j)$ on the ledger, for $j \in \mathcal{C}$.
3. **Recovery:** For every party $P_a \in \mathcal{C}$ that does not publish $\mathsf{Open}(s^a, r_a)$ in the Reveal phase, party P_j runs the Reconstruction phase of the PVSS protocol posting $\tilde{s}_j^a$ and $PROOF_j^a$ to the public ledger, for $1 \leq j \leq n$. Once $\frac{n}{2}$ valid decrypted shares are published, every party reconstructs h^{s^a}.

The final randomness is computed as $\rho = \prod_{j \in \mathcal{C}} h^{s^j}$.

Fig. 3. Protocol π_{SCRAPE}

majority, a PVSS scheme and a public ledger. SCRAPE is basically a coin tossing protocol with guaranteed output delivery (G.O.D.), meaning that an adversary cannot prevent honest parties from obtaining a correct output (*e.g.* by aborting before the execution is finished). Moreover, SCRAPE is publicly verifiable, meaning that anybody can analyse past (and current) protocol transcripts to verify that the protocol is being correctly executed. The reason we aim at guaranteed output delivery is twofold: 1. protecting against particularly adversarial behavior and 2. Tolerating non-byzantine failures in users after the commitment phase (*e.g.* power outage). When used to bootstrap blockchain based consensus protocols such as in [KKR$^+$16], π_{SCRAPE} has to tolerate adversaries that can force a temporary loss of consensus making the users end up with conflicting random outputs or temporarily "disconnecting" users from the network or public ledger.

We follow the general approach of [RBO89] to obtain guaranteed output delivery based on verifiable secret sharing, building on our PVSS schemes to achieve public verifiability for the final coin tossing protocol. More specifically, we use our PVSS protocols to instantiate the construction of [KKR$^+$16], which proposed to combine a PVSS scheme with a public ledger to obtain publicly verifiable G.O.D. coin tossing. The protocol is described in Figure 3. The security of π_{SCRAPE} follows from the security of the general construction of [KKR$^+$16] and the security of our protocols that was proven in the previous sections. We refer the reader to [KKR$^+$16] for a detailed discussion on the general protocol.

6 Concrete Complexity and Experiments

In this section, we discuss the concrete efficiency of our protocols. First, we present the concrete computational and communication complexity of our schemes, comparing them with the protocols of Schoenmakers [Sch99] and of Heidervand and Villar [HV09]. Next, we present experimental data from prototype implementations of our proposed protocols, comparing our protocols' performance to that of Schoenmakers protocol [Sch99].

	Distribution	Verification		Reconstruction	
	Exp.	Exp.	Pair.	Exp.	Pair.
[HV09]	$n+t$	nt	$2n$	$t+1$	$2t+1$
Protocol π_{DBS}	$2n$	n	$2n$	$t+1$	$2t+1$
[Sch99]	$4n+t$	$nt+4n$	-	$5t+3$	-
Protocol π_{DDH}	$4n$	$5n$	-	$5t+3$	-

Table 1. Concrete computational complexity in terms of numbers of exponentiations (Exp.) and pairings (Pair.) needed for each phase, considering that n shares are generated and t shares are used in reconstruction.

Computational Complexity: We first start by discussing the computational complexity of our PVSS protocols, which is compared to that of the protocols of [Sch99,HV09] in terms of numbers of exponentiations and pairings required for each phase in Table 6. Notice that the main improvement of our protocols in comparison with previous works lies in the verification phase, where π_{DBS} requires $n(t-1)$ less exponentations than the protocol of [HV09] and π_{DDH} requires nt less exponentiations than the protocol of [Sch99], where n is the number of shares and t is the threshold.

The distribution phase of π_{DBS} requires $n-t$ more exponentiations than the distribution phase of the protocol of [HV09] and $2n+t$ less exponentiations than the protocol of [Sch99]. Protocol π_{DDH} requires t less exponentiations than the protocol of [Sch99]. The smaller number of exponentiations required by π_{DBS} has a strong impact in improving the efficiency of this protocol (as we shall see in the experimental data). The reconstruction phases of π_{DBS} (resp. π_{DDH}) and the protocol of [HV09] (resp. the protocol of [Sch99]) are identical.

Notice that for our randomness beacon application we need $t = n/2$, which translates into an extra overhead of $n^2/2$ exponentiations required for the verification phase of previous protocols. Our protocols eliminate this quadratic overhead, resulting in much better scalability. For example, if 10000 users run SCRAPE based on [Sch99], 50004000 exponentiations are required in the verification phase, while our instantiation of SCRAPE would require only 50000 exponentiations, achieving a theoretical performance gain of over 100 times (though the practical performance gain is smaller due to the overhead of other operations such as polynomial arithmetics and I/O).

Communication Complexity: In Table 2, we present a comparison of the communication complexity of our schemes communication complexity with the protocols of [Sch99,HV09] in terms of number of group and ring elements required for each phase. Our DDH based scheme requires $2n$ group elements and $n+1$ ring elements to be published by the dealer while our pairings based construction requires $2n$ source group elements. In previous DDH based constructions $n+t$ group elements and $n+1$ ring elements are required, while previous pairing based constructions require $n+t$ source group elements. We argue that this difference is not significant for our randomness beacon application, since $n = t/2$ in this scenario, meaning that our schemes require an extra communication overhead of only $0.5n$ group elements.

	Distribution		Reconstruction	
	$\mathbb{G}$	$\mathbb{Z}_p$	$\mathbb{G}$	$\mathbb{Z}_p$
[HV09]	$n+t$	0	t	0
Protocol π_{DBS}	$2n$	0	t	0
[Sch99]	$n+t$	$n+1$	t	$t+1$
Protocol π_{DDH}	$2n$	$n+1$	t	$t+1$

Table 2. Concrete communication complexity in terms of numbers of elements of $\mathbb{G}$ and elements of $\mathbb{Z}_p$ needed for each phase, considering that n shares are generated and t shares are used in reconstruction (notice that no communication is needed for the reconstruction phase).

6.1 Experiments

In order to evaluate the concrete performance of our proposed protocols, we have conducted experiments with prototype Haskell implementations [Han17,Ryb17] of π_{DDH}, π_{DBS} and the PVSS scheme of [Sch99]. The implementations of π_{DDH} and the protocol of [Sch99] are based on curve P256R1 while the implementation of π_{DBS} is based on the mcl library [Mit15], which implements pairings based on the 256-bit Barreto-Naehrig curve [BN06] Fp254BNb using parameters and algorithms proposed in [AKL+11].

Notice that a version of π_{DBS} over asymmetric pairings was implemented in order to achieve better efficiency while preserving security guarantees. It is possible to instantiate π_{DBS} over an asymmetric bilinear group $\Lambda := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ under the co-DBS assumption (*i.e.* assuming that the DBS assumption holds in both source groups). This can be done by sampling $h \leftarrow \mathbb{G}_1$, $g, g' \leftarrow \mathbb{G}_2$ such that $pk_i = h^{sk_i} \in \mathbb{G}_1$ and $v_i \in \mathbb{G}_2$, adding $pk'_i = g^{sk_i} \in \mathbb{G}_2$ to the setup phase and defining the secret as $e(h^s, g')$. Now the check of validity of decrypted shares in the reconstruction phase can be done using pk'_i instead of pk_i by checking that $e(\tilde{s}_i, pk'_i) = e(\hat{s}_i, g)$. Alternatively, this check can also be done using v_i without the need for pk'_i by checking that $e(\tilde{s}_i, g) = e(h, v_i)$, though requiring each party to save all v_i until the reconstruction phase.

We analyze the execution time of each phase of Protocol π_{DDH}, Protocol π_{DBS} and the protocol of [Sch99] when processing n shares, for n from 1000 to 10000. We set $t = \frac{n}{2}$, since that's the threshold used in the SCRAPE randomness beacon application. In the case of the Generation and Verification phases, we analyze the execution time of generating and verifying n shares, while for the Reconstruction phase we analyze the execution time for decrypting and verifying validity of t shares and then using them for reconstructing the secret. The experiments were run on a machine with a Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz and 16 GB of RAM running the 4.4.0-22-generic #40-Ubuntu SMP Linux kernel.

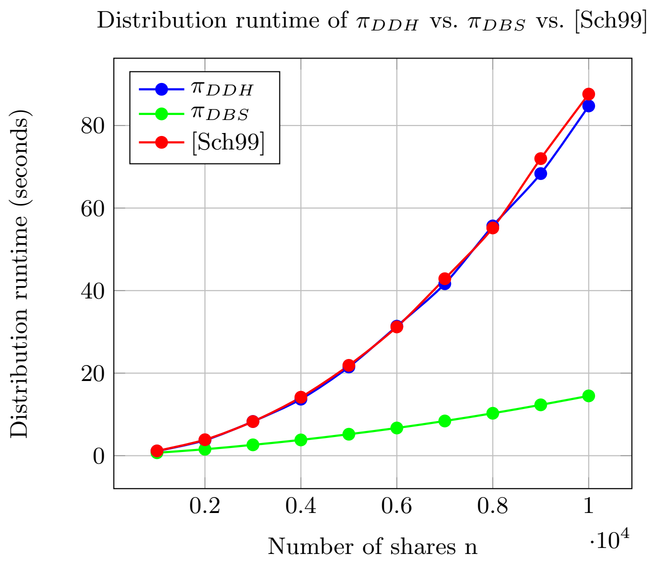

Fig. 4. Execution time of the Distribution phases of π_{DDH} vs. π_{DBS} vs. Schoenmakers' PVSS [Sch99] for a number of shares n from 1000 to 10000 and threshold $t = \frac{n}{2}$.

Distribution Phase: Even though the distribution phases of π_{DDH}, π_{DBS} and the protocol of [Sch99] are very similar, π_{DDH} requires t less exponentiations than the protocol of [Sch99] and π_{DBS} requires $2n + t$ less exponentiations than the protocol of [Sch99]. Our experimental data shows that the small number of exponentiations saved by π_{DDH} does not have a high impact in concrete performance while the much smaller number of exponentiations of π_{DBS} results in a clear efficiency improvement. The execution time of the distribution phases of π_{DDH}, π_{DBS} and the protocol of [Sch99] are compared in Figure 4.

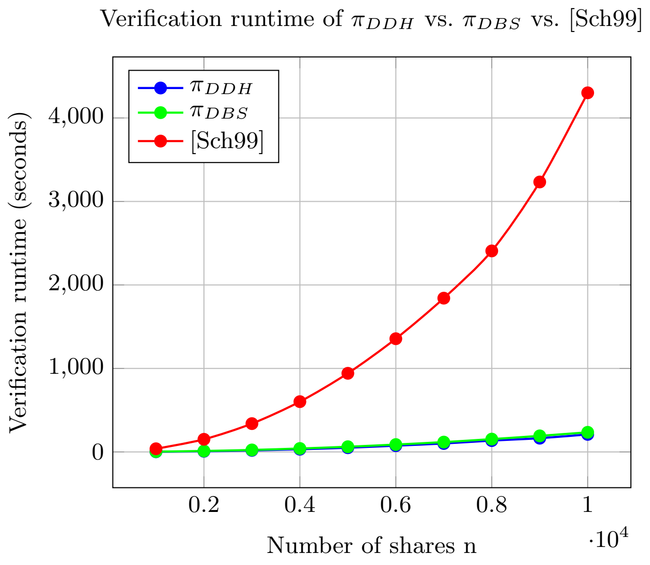

Fig. 5. Execution time of the Verification phases of π_{DDH} vs. π_{DBS} vs. Schoenmakers' PVSS [Sch99] for a number of shares n from 1000 to 10000 and threshold $t = \frac{n}{2}$.

Verification Phase: The main improvement of π_{DDH} and π_{DBS} over the protocol of [Sch99] is the verification phase that saves on nt exponentations, which amounts to saving $\frac{n^2}{2}$ exponentiations when $t = \frac{n}{2}$, as in our experiments. Figure 5 compares the execution time of the verification phases of π_{DDH}, π_{DBS} and the protocol of [Sch99]. Notice that in the case of $n = 10000$ and $t = 5000$, our schemes are more than 20 times faster than the scheme of [Sch99]. The scale in Figure 5 makes it look like π_{DDH} and π_{DBS} have the same execution time in the verification phase. However, π_{DBS} does have an overhead in comparison to π_{DDH} because it is based on pairings. Interestingly, this overhead is under 30% for $n \geq 3000$ and under 20% for $n > 5000$. We illustrate the overhead of π_{DBS} in comparison to π_{DDH} in Figure 6, where the verification pahse execution times of only our protocols are compared.

Reconstruction Phase: The reconstruction phase of π_{DDH} and the protocol of [Sch99] are exactly the same, while the reconstruction phase of π_{DBS} requires

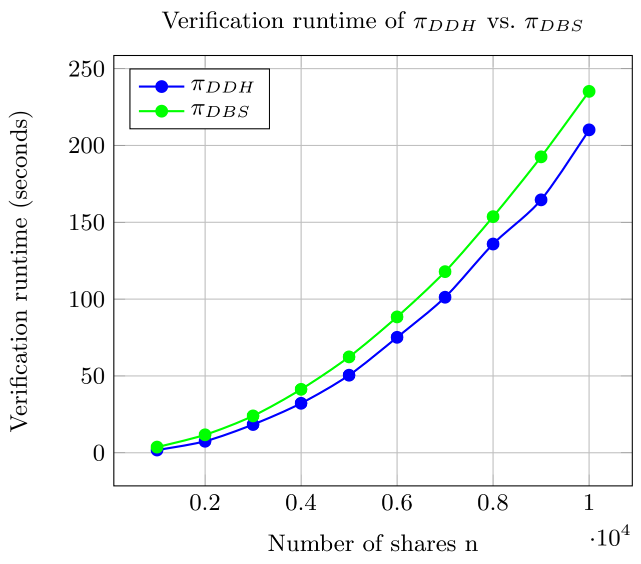

Fig. 6. Execution time of the Verification phases of π_{DDH} vs. π_{DBS} for a number of shares n from 1000 to 10000 and threshold $t = \frac{n}{2}$.

$2n$ pairing operation in comparison to generating and checking n DLEQ proofs. In these experiments, we consider the execution time of decrypting t shares, generating and verifying proofs that these t decrypted shares are valid and using them to interpolate the final secret. We can see in the experimental data that the pairing operations do have an overhead but that this overhead between 40% and 60% in comparison to π_{DDH} and the protocol of [Sch99]. The experimental data is compared in Figure 7.

7 Conclusion

We have introduced the first (n, t)-threshold PVSS scheme where only $O(n)$ public key operations are required throughout the protocol. Our main technique is a new information theoretical verification phase and we can use it to construct schemes in the ROM secure under the DDH assumption and in the plain model with pairings under the DBS assumption. This efficient PVSS scheme enables SCRAPE, a scalable protocol that implements a random beacon given an honest majority. Requiring $O(n)$ public key operations for verification still translates into $O(n^2)$ public operations per party in SCRAPE (when verifying all n shares from all n parties). Thus it is an interesting open problem to construct a similar PVSS scheme that requires only a sublinear number of public key operations. Moreover, our verification technique requires the dealer to publish at least $2n$ group elements per n shares, which could potentially be reduced to $n + t$ group

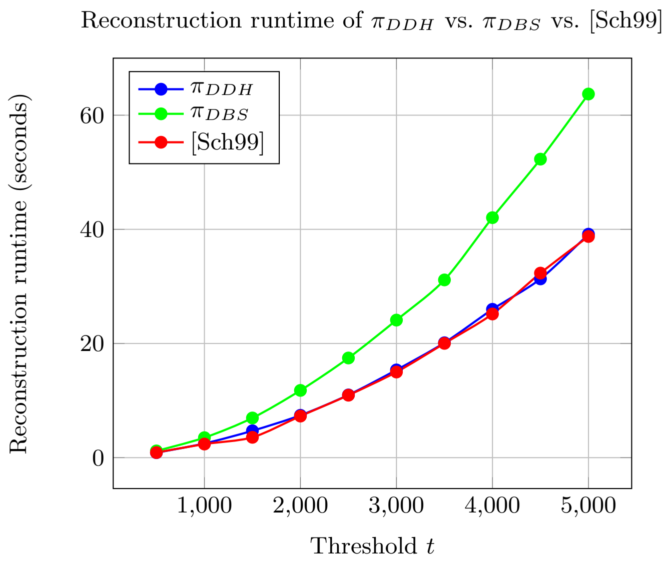

Fig. 7. Execution time of the Reconstruction phases of π_{DDH} vs. π_{DBS} vs. Schoenmakers' PVSS [Sch99] comprising decryption, decrypted share verification and interpolation using t shares as input for a threshold t from 500 to 5000.

elements as in previous works. We analyse our schemes in the stand-alone setting, leaving a composable construction as a future work.

Acknowledgements

We thank Vincent Hanquez for implementing π_{DDH} and Schoenmakers' protocol [Sch99] and Andrzej Rybczak for implementing π_{DBS}. We also thank them both for their assistance during the benchmark process.

References

Adi08. Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 335–348. USENIX Association, 2008.

AHO16. Masayuki Abe, Fumitaka Hoshino, and Miyako Ohkubo. Design in type-I, run in type-III: Fast and scalable bilinear-type conversion using integer programming. In Robshaw and Katz [RK16], pages 387–415.

AKL$^+$11. Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 48–68. Springer, Heidelberg, May 2011.

B$^+$14. Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

BCG15. Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015, 2015. `http://eprint.iacr.org/2015/1015`.

BDF$^+$15. Thomas Baignères, Cécile Delerablée, Matthieu Finiasz, Louis Goubin, Tancrède Lepoint, and Matthieu Rivain. Trap me if you can – million dollar curve. Cryptology ePrint Archive, Report 2015/1249, 2015. `http://eprint.iacr.org/2015/1249`.

BDO14. Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 175–196. Springer, Heidelberg, September 2014.

BGM16. Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In Clark et al. [CMR$^+$16], pages 142–157.

BLMR14. Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. *SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

BLN16. Daniel J Bernstein, Tanja Lange, and Ruben Niederhagen. Dual ec: a standardized back door. In *The New Codebreakers*, pages 256–281. Springer, 2016.

Blu81. Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.

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

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

BT99. Fabrice Boudot and Jacques Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99*, volume 1726 of *LNCS*, pages 87–102. Springer, Heidelberg, November 1999.

CDD$^+$16. Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, and Jesper Buus Nielsen. Rate-1, linear time and additively homomorphic UC commitments. In Robshaw and Katz [RK16], pages 179–207.

CDE$^+$16. Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains - (A position paper). In Clark et al. [CMR$^+$16], pages 106–125.

CGMA85. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th FOCS*, pages 383–395. IEEE Computer Society Press, October 1985.

Cle86. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 364–369. ACM, 1986.

CMR^{+}16. Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors. *FC 2016 Workshops*, volume 9604 of *LNCS*. Springer, Heidelberg, February 2016.

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

DMS04. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

DPSW16. Jean Paul Degabriele, Kenneth G. Paterson, Jacob C. N. Schuldt, and Joanne Woodage. Backdoors in pseudorandom number generators: Possibility and impossibility results. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 403–432. Springer, Heidelberg, August 2016.

Fel87. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437. IEEE Computer Society Press, October 1987.

FO98. Eiichiro Fujisaki and Tatsuaki Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 32–46. Springer, Heidelberg, May / June 1998.

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

GKL15. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, April 2015.

GPS06. S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/2006/165`.

GRFJ14. Mainak Ghosh, Miles Richardson, Bryan Ford, and Rob Jansen. A torpath to torcoin: proof-of-bandwidth altcoins for compensating relays. Technical report, DTIC Document, 2014.

Han17. Vincent Hanquez. pvss-haskell, 2017. `https://github.com/input-output-hk/pvss-haskell`.

HV09. Somayeh Heidarvand and Jorge L. Villar. Public verifiability from pairings in secret sharing schemes. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 294–308. Springer, Heidelberg, August 2009.

Jha11. Mahabir Prasad Jhanwar. A practical (non-interactive) publicly verifiable secret sharing scheme. In Feng Bao and Jian Weng, editors, *Information Security Practice and Experience - 7th International Conference, ISPEC 2011, Guangzhou, China, May 30 - June 1, 2011. Proceedings*, volume 6672 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2011.

JVSN14. Mahabir Prasad Jhanwar, Ayineedi Venkateswarlu, and Reihaneh Safavi-Naini. Paillier-based publicly verifiable (non-interactive) secret sharing. *Designs, Codes and Cryptography*, 73(2):529–546, 2014.

KKR$^+$16. Aggelos Kiayias, Ioannis Konstantinou, Alexander Russell, Bernardo David, and Roman Oliynykov. A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889, 2016. `http://eprint.iacr.org/2016/889`.

KMS$^+$16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.

LV08. Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 360–379. Springer, Heidelberg, March 2008.

LW15. Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015. `http://eprint.iacr.org/2015/366`.

Mau96. Ueli M. Maurer, editor. *EUROCRYPT'96*, volume 1070 of *LNCS*. Springer, Heidelberg, May 1996.

Mit15. Shigeo Mitsunari. mcl, 2015. `https://github.com/herumi/mcl`.

MS81. Robert J. McEliece and Dilip V. Sarwate. On sharing secrets and reed-solomon codes. *Commun. ACM*, 24(9):583–584, 1981.

Nak08. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

Nao91. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Maurer [Mau96], pages 387–398.

Rab83. Michael O. Rabin. Transaction protection by beacons. *J. Comput. Syst. Sci.*, 27(2):256–267, 1983.

RBO89. Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.

RK16. Matthew Robshaw and Jonathan Katz, editors. *CRYPTO 2016, Part III*, volume 9816 of *LNCS*. Springer, Heidelberg, August 2016.

RV05. Alexandre Ruiz and Jorge Luis Villar. Publicly verfiable secret sharing from paillier's cryptosystem. In Christopher Wolf, Stefan Lucks, and Po-Wah Yau, editors, *WEWoRC 2005 - Western European Workshop on Research in Cryptology, July 5-7, 2005, Leuven, Belgium*, volume 74 of *LNI*, pages 98–108. GI, 2005.

Ryb17. Andrzej Rybczak. pvss-haskell, 2017. `https://github.com/arybczak/pvss-haskell`.

Sch99. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 148–164. Springer, Heidelberg, August 1999.

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

SJK$^+$16. Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. Cryptology ePrint Archive, Report 2016/1067, 2016. `http://eprint.iacr.org/2016/1067`. To appear at IEEE Security & Privacy 2017.

Sta96. Markus Stadler. Publicly verifiable secret sharing. In Maurer [Mau96], pages 190–199.

SV15. Berry Schoenmakers and Meilof Veeningen. Universally verifiable multiparty computation from threshold homomorphic cryptosystems. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 3–22. Springer, Heidelberg, June 2015.

vdHLZZ15. Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 137–152, New York, NY, USA, 2015. ACM.

WCGFJ12. David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pages 179–192, Berkeley, CA, USA, 2012. USENIX Association.