Batch-verifiable Secret Sharing with Unconditional Privacy

Stephan Krenn, Thomas Lorünser and Christoph Striecks
AIT Austrian Institute of Technology, Vienna, Austria
{*stephan.krenn, thomas.loruenser, christoph.striecks*}*@ait.ac.at*

Keywords: Unconditional Privacy, Verifiable Secret Sharing, Batch Verification.

Abstract: We propose the first batch-verifiable secret sharing scheme with a significant security property, namely that of unconditional privacy. Verifiability and privacy of secret-shared messages are a crucial feature, e.g., in distributed computing scenarios, and verifiable secret sharing schemes with unconditional privacy (but without a batching feature) exist for a long time, e.g., Ben-Or, Goldwasser, and Wigderson (STOC 1988). Unfortunately, those schemes are able to verify only a single message at a time which, however, is not a very realistic scenario in a more practical setting. Namely, large files in real-world implementations are often split into many message blocks on a several-byte level and, thus, many known single-message verifiable secret sharing schemes tend to behave inefficiently in such a scenario. To improve practicability, batch-verifiable secret sharing was proposed by Bellare, Garay, and Rabin (ACM PODC 1996). In their scheme, the servers are able to verify a batch of messages (instead of only one) at almost the same amortized efficiency costs in comparison to efficient existing verifiable secret sharing schemes that only deal with single messages. However, the Bellare-Garay-Rabin scheme does not consider the important security property of unconditional privacy. Unconditionally private schemes information-theoretically guarantee privacy even against computationally unbounded adversaries and, hence, can be seen to be private in a long-term sense. In this work, we lift the Bellare-Garay-Rabin scheme to the unconditional privacy setting in a rigorous manner while preserving the practicability of their scheme simultaneously.

1 INTRODUCTION

Secret sharing is one of the most fundamental information-theoretically secure cryptographic primitives. It allows a party (the *dealer*) to split a message M into a multitude of *shares* (say n pieces) such that only designated *qualified sets* of shares are able to reconstruct M. At the same time, no other set of shares is able to infer any information about the message in an information-theoretic sense.

Secret sharing was simultaneously introduced by Shamir (Shamir, 1979) and Blakely (Blakley, 1979). Both works consider a threshold setting where, for some $t \leq n$, any subset of at least $t+1$ of the n shares is qualified while any set of at most t shares is not. Fundamentally, secret sharing is used as a building block in many cryptographic areas, such as within secure multi-party computation (e.g., (Chaum et al., 1987)) or broadcast encryption (e.g., (Naor and Pinkas, 2001)). In a pure practical sense, real-world implementations of secret sharing are given in a variety of products, such as in a Symantec PGP Desktop solution (Symantic, 2012), in the Ubuntu libraries (gfshare, 2010), within CHARM (Akinyele et al., 2013), or ARCHISTAR (Lorünser et al., 2015).

While plain secret sharing guarantees availability (at least for appropriately chosen values of n and t) and confidentiality in the case of up to t corrupt share holders, it does not give any guarantees in case of a corrupt dealer. However, this might often be required, e.g., in the case of data sharing, where multiple, mutually untrusted dealers are allowed to write data. In this case, it becomes necessary to guarantee that the shares written by a dealer are actually consistent. This is exactly what can be guaranteed by *verifiable secret sharing* (VSS) schemes which were introduced by Chor, Goldwasser, Micali, and Awerbuch (Chor et al., 1985). Within this setting, the share holders are assumed to be active, rather than being passive storage nodes only. After receiving their shares, the share holders and the dealer engage in a protocol at the end of which every honest share holder is assured that every honest qualified set of shares will be able to reconstruct the same secret. VSS has numerous applications in the fields of secure multi-party computation (Cramer et al., 1999), distributed key generation (Zhang and Zhang, 2014), distributed proxy signatures (Herranz and Sáez, 2003), audit pro-

Krenn, S., Lorünser, T. and Striecks, C.
Batch-verifiable Secret Sharing with Unconditional Privacy.
DOI: 10.5220/0006133003030311
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 303-311
ISBN: 978-989-758-209-7

tocols (Demirel et al., 2016), or Byzantine fault tolerance protocols (Happe et al., 2016). Additionally, effcient real-world implementations of (public) VSS schemes are given in the work of (D'Souza et al., 2011), where also an implementation of (Schoenmakers, 1999) is given.

One key property a VSS scheme can exhibit is (unconditional) privacy; those schemes are also called (unconditionally) private VSS schemes. Such feature allows that no information on the secret-shared message is leaked through an unqualified set of shares (unconditionally). Of course, every qualified set is able to recover this message entirely, but no other set. Starting from the unconditionally private VSS work of Ben-Or, Goldwasser, and Wigderson (Ben-Or et al., 1988), there is a variety of schemes in the cryptographic literature which focus on unconditionally (and also conditional) privacy in the VSS setting, e.g., (Rabin and Ben-Or, 1989; Gennaro et al., 2007; Patra et al., 2009a; Kumaresan et al., 2010; Feldman, 1987; Pedersen, 1991; Gennaro et al., 1998; Backes et al., 2011).

Unfortunately, many of the existing VSS schemes (and essentially all of the schemes mentioned above) are inefficient in the following sense. In the usual VSS setting, the dealer has to prove to the share holders that all their shares resulted from an honest execution of the sharing algorithm. This is essentially done by letting the dealer commit to his random coins and broadcast these commitments such that the share holders can then (potentially interactively) verify that their shares are consistent with the commitments. The mentioned scenario might be acceptable for single messages. However, this is clearly not satisfactory for a multi-message setting. For example, real-world applications usually implement secret sharing on a several-byte level, while files are often megabytes to gigabytes in size such that a file often has to be split into 2^{20} blocks (i.e., secret-shared messages) or more. As a result, the computational overhead of computing, transferring, and verifying the high number of commitments renders many existing VSS schemes impracticable in the real-world setting.

To the best of our knowledge, the only known VSS scheme which simultaneously deals with a large number of messages is that of Bellare, Garay, and Rabin (Bellare et al., 1996). Their scheme is able to verify a batch of messages with almost no extra amortized efficiency cost in comparison to existing (only single-message) VSS schemes. However, they use VSS only as a building block to achieve a more complex primitive and do not consider privacy. (Essentially, every server in their scheme learns a weighted sum over the shared batch of messages, and thus the scheme is not unconditionally private.) Since it is tempting and natural to use batch-VSS also as a standalone primitive (e.g., within the several-byte scenario mentioned above), unconditional privacy is an extremely important security feature as it allows to guarantee privacy of shared secret messages even against computationally unbounded adversaries.

In this work, we extend the batch-VSS scheme of (Bellare et al., 1996) (and also their VSS definition) to propose the first batched VSS scheme with unconditional privacy while essentially preserving the efficiency and practicability of their scheme simultaneously. Along the line, we propose rigorous VSS definitions. Although the technical step to lift Bellare et. al's result to the unconditional privacy setting is rather small, we stress that to the best of our knowledge no scientific paper has so far analyzed unconditional privacy with rigorous definitions in the batch-VSS setting. We stress that unconditional privacy is an important property of VSS schemes as it yields privacy in a long-term sense. Hence, our work can be seen as complementing the overall picture of efficient and unconditionally private batch-VSS schemes.

Motivation and Related Work. Secret sharing was simultaneously introduced in the seminal work of Shamir (Shamir, 1979) and Blakely (Blakley, 1979). Verifiable secret sharing has first been introduced by Chor et al. (Chor et al., 1985). Since then, a huge effort of work has been performed in this field. For instance, solutions that are unconditionally private and committing have been proposed for $n \geq 2t+1$ (Gennaro et al., 2001; Fitzi et al., 2006; Katz et al., 2008). If one is additionally aiming for perfect completeness, then one has to use $n \geq 3t+1$ (Ben-Or et al., 1988; Dolev et al., 1993).

For efficiency reasons, many publications consider a setting which is perfectly private but only computationally committing. The probably most prominent VSS scheme is the scheme of Pedersen (Pedersen, 1991). Its computationally private but perfectly committing counterpart has been proposed by Feldman (Feldman, 1987). An optimized generalization of Pedersen's scheme has been presented by Backes et al. (Backes et al., 2011) while Kate et al. (Kate et al., 2010) propose an efficient VSS with constant-size commitments.

All schemes mentioned (and also our proposed scheme) consider private verifiability only. That is, only the share holders are able to verify that the distributed shares are actually correct and consistent. If computations on the share holder's side are very expensive, publicly verifiable secret sharing schemes can be used, where also external parties can verify the

consistency of the distributed shares without learning anything about the shared message (Fujisaki and Okamoto, 1998; Schoenmakers, 1999; Jhanwar et al., 2014). Finally, another line of research also considers VSS in an asynchronous communication model where the adversary is allowed to partially control the network, e.g., (Canetti and Rabin, 1993; Cachin et al., 2002; Patra et al., 2009b; Backes et al., 2011).

In all works mentioned so far, verification is done on a per-message level. The only paper considering a batch version of VSS is Bellare et al. (Bellare et al., 1996) which shows how to efficiently batch-verify multiple Shamir-shared messages, but does not consider unconditional privacy. Demirel et al. (Demirel et al., 2016) suggest a protocol for auditing the servers, i.e., checking efficiently whether the distributed shares are still available at all servers. They claim that their protocol can also be used as a batch-VSS scheme if one is willing to assume that all servers behave honestly during the sharing phase, which, however, is a non-standard assumption as one typically allows up to t corrupted parties at any point of the protocol. As a result, a single corrupted server could easily carry out denial-of-service attacks in their setting.

Our Contribution. In a nutshell, we propose an efficient batch-verifiable secret sharing scheme that satisfies a strong (standard) privacy definition for VSS. We stress that former works do not focus on privacy, i.e., (Bellare et al., 1996), or achieve only a weaker correctness property, i.e., (Demirel et al., 2016).

More precisely, we first formally define what we understand by a batch-VSS scheme. While this is intuitively clear, we believe that a detailed formalization is necessary. In particular, many existing VSS definitions are often quite informal (Backes et al., 2011; Kaya and Selçuk, 2008; Pedersen, 1991; Schoenmakers, 1999) in the sense that it is not formally defined, e.g., at which points in time, the adversary is allowed to corrupt which parties and which information is then revealed. Our goal was to present strong and rigorously formal definitions for batch-VSS schemes.

In the following, we present an instantiation of our batch-VSS definitions based on the Bellare et al. scheme (Bellare et al., 1996) and, hence, on Shamir's secret sharing technique (Shamir, 1979). The instantiation is correct, unconditionally private, and statistically or computationally committing, for $n \geq 3t+1$. The batch-VSS scheme of Bellare et al. achieves the statistical committing property; however, their scheme needs a pre-processing of server secrets which our computationally committing scheme does not need. Nevertheless, we can adapt our scheme to work with the Bellare et al. setup and, thus, our statistically committing scheme achieves the same committing property. Furthermore, our instantiation is perfect, i.e., the shares stored by each server have the same size as the original message.

More technically, the dealer starts with sharing each message M_j using a degree-t polynomial $f_j(x)$ with coefficients $a_{j,v}$ according to Shamir's scheme. Further, it draws another polynomial $r(x)$ with uniform coefficients $b_0,\ldots,b_t$. After having received the shares, the servers first interactively agree on a uniformly random challenge w which is also sent to the dealer. The dealer then computes $t+1$ linear combination of the polynomial coefficients, namely $C_v := \sum_{j=1}^{m} a_{j,v} w^j + b_v$, using the coefficients b_v of $r(x)$ as blinding terms. Each server is now able to compute a linear combination of all C_v and can check this against a linear combination of its own shares. In the beginning of the reconstruction phase, each server broadcasts its shares, which enables each server to reconstruct the secret and verify the consistent sharing of the dealer.

2 PRELIMINARIES

We introduce some notations and basic cryptographic primitives. For $n \in \mathbb{N}$, let $[n] := \{1,\ldots,n\}$ and let $\lambda \in \mathbb{N}$ be the security parameter. We denote algorithms by sans-serif letters ($\mathsf{A},\mathsf{B},\ldots$) and sets by calligraphic letters ($\mathcal{R},\mathcal{S},\ldots$). For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. For an algorithm A, let $y \leftarrow \mathsf{A}(1^\lambda, x)$ be the process of running A, on input 1^λ and x, with access to uniformly random coins and assigning the result to y. We assume that all algorithms take 1^λ as input and we will sometimes not make this explicit in the following. An algorithm A is probabilistic polynomial time (PPT) if its running time is polynomially bounded in λ. Furthermore, we write $\Pr[\mathcal{E} : \Omega]$ to denote the probability of event $\mathcal{E}$ over the probability space Ω. A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if it vanishes faster than the inverse of any polynomial, i.e., if $\forall c \exists \lambda_0 \forall \lambda \geq \lambda_0 : |f(\lambda)| \leq 1/\lambda^c$. We write $(o_A; o_B; \ldots) \leftarrow \langle A(in_A); B(in_B); \ldots \rangle$ for an interactive protocol between participants A, B. (Here, in_A, in_B and o_A, o_B resp. denote the inputs and outputs.)

Commitment Schemes. A commitment scheme with message space G consists of the three PPT algorithms $(\mathsf{CPar}, \mathsf{Com}, \mathsf{Open})$ with the properties correctness (honestly computed commitments can always be opened by an honest party), hiding (a commitment must not leak any information about the

committed message), and binding (no adversary must be able to open a commitment to two different messages). $\mathsf{CPar}(1^\lambda)$, on input 1^λ in unary, outputs public parameter pp. $\mathsf{Com}(pp,M)$, on input a message $M \in G$, outputs a commitment-witness pair (C,d). $\mathsf{Open}(pp,C,M,d)$ outputs a verdict $b \in \{0,1\}$. (We refer to the full version for a formal definition.)

(Batch-verifiable) Secret Sharing. The most prominent threshold secret sharing scheme is due to Shamir (Shamir, 1979) and is based on the observation that in a field a polynomial of degree t is uniquely determined by $t+1$ function values. Let therefore $\mathbb{F}_q$ be a field with q elements. Usually, we have n server and a dealer; the dealer gets as input a message $M \in \mathbb{F}_q$ that is supposed to be secret-shared among the servers. Concretely, to share M, the dealer draws a random polynomial $f(x) = a_t x^t + \cdots + a_1 + M$ in $\mathbb{F}_q[x]$, and outputs $f(i)$ as the i-th share. To reconstruct M from at least $t+1$ values, one can now simply compute the interpolation polynomial defined by the given shares and output $f(0)$. Knowing at most t $f(x)$-values (for pairwise distinct x) does not reveal any information about the polynomial.

Bellare et al. (Bellare et al., 1996) proposed an elegant (batch-)verifiable secret sharing schemes for verifying degree-t polynomials. In their VSS construction (without the batching property), a message M is secret-shared via Shamir's scheme, i.e., the dealer chooses a degree-t polynomial $f(x) := a_{j,t}x^t + \cdots + a_{j,1} + M$ and sends $f(i)$ to server i. Additionally, the dealer shares a second polynomial $r(x)$. Further, all n servers agree on a uniformly random value w and each server broadcasts $C_i = f(i) + w \cdot r(i)$. Now, every server can interpolate a polynomial over the C_i-values and check if such a polynomial has degree $\leq t$. If this is the case, the server agrees, otherwise it rejects.

In their batched-VSS construction, for messages M_j, $j \in [m]$, the dealer Shamir-shares M_j using a degree-t polynomial $f_j(x) := a_{j,t}x^t + \cdots + a_{j,1} + M_j$. It then sends $(f_j(i))_{j\in[m]}$ to the server S_i. Next, the servers engage in a "joint randomness computation" protocol to obtain a random value w. Further, each server computes a linear combination of its shares $f_j(i)$ as $C_i := \sum_{j=1}^m f_j(i) \cdot w^j$ and broadcasts C_i. Using these values, every S_i then efficiently interpolate a polynomial using the C_i-values (since $n \geq 3t+1$) and verify that the interpolated polynomial has degree $\leq t$. If this is the case, S_i accepts, otherwise it rejects. (Note that already $t+1$ broadcasted values C_i reveal enough information.) However, each server learns the weighted sum $\sum_{j=1}^m f_j(i) \cdot w^j$ of all values C_i after the broadcast. Hence, unconditional privacy of the secret-shared messages is not given.

3 THE SECURITY MODEL OF BVSS

In this section, we introduce the syntax and security requirements of batch-verifiable secret sharing.

3.1 Syntax

An (n,m,t)-batch-verifiable secret sharing scheme BVSS (dubbed (n,m,t)-BVSS for short) is a system consisting of a *dealer* D and n servers $\mathsf{S}_1,\ldots,\mathsf{S}_n$. Besides the number of servers, it is parametrized by the reconstruction threshold $t+1 \in \mathbb{N}, 1 \leq t < n$, the number m of messages that can be batch-verified, the message space $\mathcal{M}$, and the share space $\mathcal{S}$.

An (n,m,t)-BVSS consists of a *parameter generation* algorithm SPar that generates public parameters which are accessible to all participants. Depending on the concrete instantiation, it might be necessary that the public parameters are generated by a trusted third party, or through a joint computation of the different system participants. Furthermore, the system consists of two interactive protocols (or *phases*). During an invocation of the *sharing phase*, the dealer can share m messages $M_j \in \mathcal{M}$ among the servers $\mathsf{S}_1,\ldots,\mathsf{S}_n$ in a confidential yet provably consistent way. Then, in the *reconstruction phase*, the server jointly reconstruct the shared messages by broadcasting their shares and executing an algorithm Rec. It is required that all algorithms and protocols are PPT. Furthermore, we assume that each server has access to fresh and uniformly random random coins whenever necessary.

An (n,m,t)-BVSS is defined by the following algorithms and phases:

Parameter Generation. $\mathsf{SPar}(1^\lambda)$, on input unary 1^λ, outputs public parameters pp. (We assume that all participants have implicitly access to pp.)

Sharing Phase. In the beginning of this phase, the dealer D, on input pp and messages $(M_1,\ldots,M_m) \in \mathcal{M}^m$, obtains shares $(s_{1,j},\ldots,s_{n,j}) \in \mathcal{S}^n$ of M_j, for all $j \in [m]$. Further, D distributes $(s_{i,1},\ldots,s_{i,m})$ to S_i, for all $i \in [n]$. In each round, each server can communicate with every other server privately and is eligible to broadcast data. After the last round, each server S_i outputs its shares and a (potentially empty) auxiliary parameter as $\vec{s}_i = (s_{i,1},\ldots,s_{i,m},aux_i)$,for $i \in [n]$, while the dealer D outputs ε.

Reconstruction Phase. In the beginning of this phase, each server S_i broadcasts $\vec{s}_i$, for $i \in [n]$. Further, the deterministic reconstruction algorithm $\mathsf{Rec}(pp,\vec{s}_1,\ldots,\vec{s}_n)$ outputs messages

$(M_1,\ldots,M_m) \in (\mathcal{M} \cup \{\perp\})^m$ which is also defined to be the protocol's output.

3.2 Security Requirements

In the following, we define the formal security guarantees that need to be fulfilled by a BVSS scheme. Even though these requirements are very similar to the standard security definitions for VSS, we will discuss them in detail. This is because in the literature, the security of VSS schemes are often formulated in a rather informal way (Backes et al., 2011; Kaya and Selçuk, 2008; Schoenmakers, 1999), which might potentially cause subtle differences in the way they are interpreted.

From now on, we will say that an adversary is *t-valid* if and only if it corrupts at most t servers $\mathsf{S}_{i_1},\ldots,\mathsf{S}_{i_t}$. We therefore grant the adversary access to a corruption oracle $\mathsf{O}_{\mathsf{cor}}$ in all security experiments. At any point in these experiments, the adversary may ask $\mathsf{O}_{\mathsf{cor}}$ to corrupt a server of its choice, receiving its entire internal view (including all messages received, its share, etc.) as a response. A corrupted server is then under full control of A, who can, e.g., send or broadcast data on behalf of the server.

We continue with defining the correctness, privacy, and commitment properties of an (n,m,t)-BVSS scheme.

Correctness. Informally, *correctness* (or, sometimes also dubbed completeness) says that for an honest dealer, it is always guaranteed that the original messages $M_1,\ldots,M_m$ can be recovered from the shares accepted by the servers — even if up to t servers behaved arbitrarily maliciously during the sharing phase. An (n,m,t)-BVSS scheme is *correct* if for every t-valid PPT adversary A there exists a negligible function negl such that the following holds true: For all $M_1,\ldots,M_m \in \mathcal{M}, pp \leftarrow \mathsf{SPar}(1^\lambda), (\varepsilon;\vec{s}_1;\ldots;\vec{s}_n) \leftarrow \langle \mathsf{D}(pp,(M_j)_{j\in[m]});\mathsf{S}_1(pp);\ldots;\mathsf{S}_n(pp)\rangle$, the probability that $\mathsf{Rec}(pp,\vec{s}_1,\ldots,\vec{s}_n) = (M_1,\ldots,M_n)$ holds is at least $1-\mathsf{negl}(\lambda)$. We say that a BVSS scheme is perfectly correct if $\mathsf{negl}(\lambda) = 0$, for all λ.

Note here that the output of corrupted servers can be arbitrarily chosen by the adversary. In the literature, correctness is sometimes only required if the dealer and all servers are honest, e.g., (Demirel et al., 2016; Stadler, 1996). However, we believe this is a too weak requirement for practical purposes as it would enable any single malicious server to launch denial-of-service attacks. It is thus important that correctness also gives robustness guarantees against malicious servers.

Privacy. A batch verifiable secret sharing scheme is called *private* (i.e., satisfies *privacy*) if no adversary controlling up to t servers can learn any information about the message distributed by the dealer. The privacy experiment, dubbed $\mathsf{Exp}^{\text{privacy}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda)$, in which the adversary has access to $\mathsf{O}_{\mathsf{cor}}$, is defined as follows: First, fresh public parameters $pp \leftarrow \mathsf{SPar}(1^\lambda)$ are honestly generated and passed to A. Then, A chooses messages tuples $(M_{0,1},\ldots,M_{0,m}),(M_{1,1},\ldots,M_{1,m}) \in \mathcal{M}^m \times \mathcal{M}^m$ and sends them to the experiment. Next, the experiment tosses a coin $b \leftarrow \{0,1\}$ and executes the sharing phase on one of the message tuples, i.e., it launches the following protocol, taking the roles of the dealers and all honest servers: $(\varepsilon;\vec{s}_1;\ldots;\vec{s}_n) \leftarrow \langle \mathsf{D}(pp,(M_{b,j})_{j\in[m]});\mathsf{S}_1(pp);\ldots;\mathsf{S}_n(pp)\rangle$. Eventually, A outputs a guess b'; if $b = b'$ holds, then the experiment outputs 1, otherwise the experiment outputs 0.

An (n,m,t)-BVSS protocol satisfies *privacy* if and only if for any t-valid PPT adversary A there exists a negligible function negl such that the following holds true: $\left|\Pr\left[\mathsf{Exp}^{\text{privacy}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda) = 1\right] - 1/2\right| \leq \mathsf{negl}(\lambda)$. We say that a BVSS is unconditionally private if $\mathsf{negl}(\lambda) = 0$, for all λ.

Commitment. Finally, the *commitment* property is the distinguishing feature of verifiable secret sharing schemes. It guarantees that the dealer cannot change its mind about the distributed value after the sharing phase — even if it adaptively corrupts up to t servers. That is, if one has access to all shares, then one will always reconstruct the very same secret messages $M'_j \in \mathcal{M}$, even if up to t shares are arbitrarily and adaptively modified by the adversary. Consider the following commitment experiment, dubbed $\mathsf{Exp}^{\text{commit}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda)$, in which the adversary has access to $\mathsf{O}_{\mathsf{cor}}$: First, fresh public parameters $pp \leftarrow \mathsf{SPar}(1^\lambda)$ are honestly generated and passed to A. The experiment runs the sharing phase $\langle \mathsf{D}(pp,(M_j)_{j\in[m]});\mathsf{S}_1(pp);\ldots;\mathsf{S}_n(pp)\rangle$ for $M_1,\ldots,M_m \in \mathcal{M}$ chosen by A. Let $(\vec{s}_i)_{i\notin\mathcal{C}}$ be the shares of the non-corrupted parties. In the next step, the adversary is given $(\vec{s}_i)_{i\notin\mathcal{C}}$ as input and outputs two full sets of shares $(\vec{s}_i)_{i\in[n],0},(\vec{s}_i)_{i\in[n],1}$, such that $\#\{i \in \mathcal{C} : \vec{s}_i = \vec{s}_{i,0}\} \geq n-t$ and similar for $\vec{s}_{i,1}$. The experiment outputs 1, if and only if $\mathsf{Rec}(\vec{s}_{1,0},\ldots,\vec{s}_{n,0}) \neq \mathsf{Rec}(\vec{s}_{1,1},\ldots,\vec{s}_{n,1})$.

An (n,m,t)-BVSS protocol is now said to be *committing* if and only if for every t-valid adversary A there exists a negligible function negl such that the following holds true: $\Pr\left[\mathsf{Exp}^{\text{commit}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$. In particular, we say that a BVSS is unconditionally committing if $\mathsf{negl}(\lambda) = 0$, for all λ, and computational committing otherwise.

Let us explain the rationale behind the commit-

ment definition. Namely, the adversary can already compromise a number of servers before and during the sharing phase. Then, in the reconstruction phase, all servers broadcast their shares. The adversary is thus allowed to see all shares held by honest servers, and may adaptively corrupt further ones. (Alternatively, one could have required that the adversary must decide which servers to corrupt before the shares were revealed; however, we believe that our stronger modeling is reasonable as it guarantees security against adaptive adversaries that might be able to, e.g., block broadcast messages.) The requirement now is that no matter how the adversary modifies the shares of corrupted parties, the reconstruction phase will always yield the same result and, thus, the dealer is committed to this value after the sharing phase.

4 AN EFFICIENT AND UNCONDITIONALLY PRIVATE INSTANTIATION OF BVSS

In this section, we present an efficient and unconditionally private instantiation of an (n,m,t)-BVSS scheme and prove that it indeed satisfies the security requirements from Sec. 3.

4.1 An (n,m,t)-BVSS Instantiation

In the following, we let p be a prime or a prime power and define the message space of our BVSS scheme as $\mathcal{M} := \mathbb{F}_p$. Furthermore, we use an interactive randomness generation protocol $(\mathsf{RPar}, \mathsf{Rnd}_{\mathsf{BVSS}})$ as a building block. Informally, randomness generation protocol, executed among the n servers, has to guarantee that all honest servers and the dealer receive the same uniformly random output from $\mathbb{F}_p$, even if up to t of the servers behaved maliciously. For a formal definition and a concrete instantiation of a randomness generation protocol, we refer to Sec. 4.2.

Perfect (n,m,t)-BVSS for $n \geq 3t+1$. We now present an instantiation which works for all $n \geq 3t+1$ and is perfect in the sense that it only has only a constant additive storage overhead compared to plain Shamir secret sharing. (The scheme is a variant of the BVSS scheme presented in (Bellare et al., 1996).) The algorithms and protocols of our (n,m,t)-BVSS scheme are defined as follows:

Parameter Generation. $\mathsf{SPar}(1^\lambda)$ obtains $pp' \leftarrow \mathsf{CPar}(1^\lambda)$ as well as $pp'' \leftarrow \mathsf{RPar}(1^\lambda, n, t, p)$, and outputs public parameters $pp := (pp', pp'')$.

Sharing Phase. The sharing phase consists of the following two rounds plus the round(s) needed in the joint randomness generation protocol:

(a) The dealer D, on input pp and $(M_1, \ldots, M_m) \in \mathbb{Z}_p^m$, chooses $m+1$ degree-t polynomials

$$f_j(x) = a_{j,t}x^t + \cdots + a_{j,1}x + M_j$$
$$\text{and} \quad r(x) = b_t x^t + \cdots + b_0$$

with (uniform) coefficents $a_{j,v}, b_v, b_0 \leftarrow \mathbb{Z}_p$, for all $v \in [t]$ and all $j \in [m]$. Further, the dealer D sends shares $((f_j(i))_{j\in[m]}, r(i))$ to each server S_i, for all $i \in [n]$.

(b) Once every server S_i received $((\hat{f}_{j,i})_{j\in[m]}, \hat{r}_i)$ from the dealer D, the servers and the dealer engage in an execution of the joint randomness generation protocol. Let w be the output of $\mathsf{Rnd}_{\mathsf{BVSS}}(1^\lambda, n, t, p)$. (Note that after this step, by definition of the building block, it is guaranteed that all honest S_i and D hold the same uniformly random value w.)

(c) Next, D computes $C_v := \sum_{j=1}^m a_{j,v}w^j + b_v$, for all $v \in [t] \cup \{0\}$, where $a_{j,0} := M_j$, and broadcasts $(C_v)_{v\in[t]\cup\{0\}}$. After the broadcast, D outputs ε.

(d) Upon having received $\hat{C}_0, \ldots, \hat{C}_t$, each server S_i verifies the consistency of its shares by validating that

$$\hat{C}_0 + \hat{C}_1 i + \cdots + \hat{C}_t i^t \stackrel{?}{=} \sum_{j=1}^m \hat{f}_{j,i}w^j + \hat{r}_i \qquad (1)$$

holds. (Note that each S_i has received the values $((\hat{f}_{j,i})_{j\in[m]} = f_j(i)_{j\in[m]}, \hat{r}_i = r(i))$ as described in step (b).) If the Eq. (1) holds, then S_i outputs $\vec{s}_i := (\hat{f}_{j,i})_{j\in[m]}$; otherwise, it terminates the protocol.

Reconstruction Phase. The reconstruction procedure is only one-round. Concretely, each server first broadcasts its shares to all other servers. Now, on input $(\vec{s}_1, \ldots, \vec{s}_n)$, Rec first verifies each $\vec{s}_i$ by using a Berlekamp-Welch decoder (Welch and Berlekamp, 1983). If less than $2t+1$ values $(\vec{s}_i)_{i\in[n]}$ are valid decodings, Rec outputs $\perp$. Otherwise, for each $j \in [m]$, it takes the respective parts of the valid shares and computes the degree-t interpolation polynomial f'_j, and outputs the messages $M'_j := f'_j(0)$, for all $j' \in [m]$.

Looking at our protocol specification, it can be seen that the computational overhead on the server side is independent of the batch size, except for the computation of the sum in Eq. (1). However, in practice, these costs are negligible, in particular compared to the evaluation of m equations of the same form in the direct generalization of existing VSS schemes.

4.2 Joint Randomness Generation

In our BVSS protocol, we require that n servers and the dealer jointly agree on a uniformly random value $w \in \mathbb{Z}_p$ in the sharing phase. Such a joint randomness generation scheme for $\mathbb{Z}_p$ consists of the following algorithm and protocol. The parameter generation algorithm $\mathsf{RPar}(1^\lambda, n, t, p)$ outputs public parameters pp. The randomness generation $\mathsf{Rnd}_{\mathsf{BVSS}}$ is an interactive protocol executed between n servers (where t of them might be malicious), where at the end of the protocol each server and the dealer determine an output w from $(w_1; \ldots; w_n) \leftarrow \langle \mathsf{S}_1(pp); \ldots; \mathsf{S}_n(pp) \rangle$ such that w is the majority value of $(w_1; \ldots; w_n)$; i.e., the protocol goes over all received w_j and count the occurrence of the values. If there is no single value that appears more often than other values or no values are output by the servers, the protocol outputs $\perp$. (We mention that in an instantiation below, we will assume that $n \geq t+1$; hence, the protocol will always output at least one value w_i, for some $i \in [n]$.)

Joint Randomness Generation Instantiation from any Commitments. For our instantiation, let $\mathsf{C} = (\mathsf{CPar}, \mathsf{Com}, \mathsf{Open})$ be an arbitrary commitment scheme with message space $\mathbb{Z}_p$ in the sense of Sec. 3. The parameter generation RPar and the interactive randomness generation protocol $\mathsf{Rnd}_{\mathsf{BVSS}}$ between the servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$, for $n \geq t+1$, are defined as follows: The parameter generation $\mathsf{RPar}(1^\lambda, n, t, p)$, on input 1^λ and integers $n, t, p \in \mathbb{N}$, with $n \geq t+1$, outputs $pp := (pp', n, t, p)$, for $pp' \leftarrow \mathsf{CPar}(1^\lambda)$. Within the randomness generation protocol $\mathsf{Rnd}_{\mathsf{BVSS}}$, each server S_i, on input pp, samples $w_i \leftarrow \mathbb{Z}_p$ and broadcasts a commitment C_i in the first round, for $(C_i, d_i) \leftarrow \mathsf{Com}(pp', w_i)$, for all $i \in [n]$. In the second round, each server S_i broadcasts the corresponding opening (w_i, d_i), for all $i \in [n]$. Let $w'_1, \ldots, w'_l$ be the received consistent openings, for $l \in [n]$. Each server S_i outputs $w_i := w'_1 + \cdots + w'_l \bmod p$.

Lemma 1. *The protocol above has the uniform output property.*

Proof. Due to space constraints, we refer to the full version for a proof. □

Note that an instantiation of a commitment scheme can only be perfectly binding and computationally hiding or computationally binding and perfectly hiding. Hence, with our joint randomness construction above, we can only guarantee the uniform output against computationally bounded adversaries. To address computationally unbounded adversaries in the joint randomness generation protocol, we can use the joint randomness generation protocol from (Bellare et al., 1996) using a setup with pre-processing. Their distributed randomness generation instantiation achieves the uniform output property of the joint generation algorithm unconditionally; in particular, they do not rely on commitments. Nevertheless, they assume a trusted setup with pre-sharing of server secrets in the setup phase of their protocol (although they only need the invocation of the setup once). Our scheme can be easily adapted to work with (a variant of) their joint randomness computation and, hence, we no longer have to rely on commitments. However, we then have to rely on the assumption of a trusted setup and pre-processing. (We refer to (Bellare et al., 1996) for more details.)

4.3 Theorem

Theorem 1. *If* $(\mathsf{RPar}, \mathsf{Rnd}_{\mathsf{BVSS}})$ *is a joint randomness generation protocol in the sense of Sec. 4.2, then the protocol* BVSS *as described in Sec. 4.1 is an* (n, m, t)*-BVSS protocol, for* $n \geq 3t+1$ *and for all* $m \in \mathbb{N}$ *(where all values are polynomial in* λ*).*

Proof. We omit the proof here due to space constraints and refer to the full version of the paper. □

5 CONCLUSION

In this work, we presented the first unconditionally private instantiation of a batch-verifiable secret sharing scheme based on the VSS scheme of Bellare, Garay, and Rabin (Bellare et al., 1996). Their solution is extremely efficient and introduce almost no amortized overhead in communication and storage compared to plain Shamir secret sharing when larger batches are considered, i.e., it is possible to add verifiability almost for free. However, their work do not consider unconditional privacy. With this work being a starting point, a lot of interesting research directions are conceivable for future work. For instance, generalizations of our approach to computationally private secret sharing schemes (e.g., based on the information dispersal scheme of Krawczyk (Krawczyk, 1993)) or to VSS protocols developed for an asynchronous network setting (Cachin et al., 2002) could be of practical relevance. Similarly, batching algorithms for proactive secret sharing are an interesting open problem.

ACKNOWLEDGMENT

We thank the anonymous reviewers for valuable comments. The authors were supported by the EU H2020 project PRISMACLOUD.

REFERENCES

Akinyele, J. A., Garman, C., Miers, I., Pagano, M. W., Rushanan, M., Green, M., and Rubin, A. D. (2013). Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*.

Backes, M., Kate, A., and Patra, A. (2011). Computational Verifiable Secret Sharing Revisited. In *ASIACRYPT*.

Bellare, M., Garay, J. A., and Rabin, T. (1996). Distributed Pseudo-Random Bit Generators - A New Way to Speed-Up Shared Coin Tossing. In *PODC*.

Ben-Or, M., Goldwasser, S., and Wigderson, A. (1988). Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*.

Blakley, G. R. (1979). Safeguarding cryptographic keys. AFIPS National Computer Conference.

Cachin, C., Kursawe, K., Lysyanskaya, A., and Strobl, R. (2002). Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. In *CCS*.

Canetti, R. and Rabin, T. (1993). Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *STOC*.

Chaum, D., Crépeau, C., and Damgård, I. (1987). Multiparty unconditionally secure protocols (abstract). CRYPTO.

Chor, B., Goldwasser, S., Micali, S., and Awerbuch, B. (1985). Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). FOCS.

Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., and Rabin, T. (1999). Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *EUROCRYPT*.

Demirel, D., Krenn, S., Lorünser, T., and Traverso, G. (2016). Efficient Third Party Auditing for a Distributed Storage System. In *ARES*.

Dolev, D., Dwork, C., Waarts, O., and Yung, M. (1993). Perfectly Secure Message Transmission. *Journal of the ACM*.

D'Souza, R., Jao, D., Mironov, I., and Pandey, O. (2011). Publicly verifiable secret sharing for cloud-based key management. INDOCRYPT.

Feldman, P. (1987). A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *FOCS*.

Fitzi, M., Garay, J. A., Gollakota, S., Rangan, C. P., and Srinathan, K. (2006). Round-Optimal and Efficient Verifiable Secret Sharing. In *TCC*.

Fujisaki, E. and Okamoto, T. (1998). A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications. In *EUROCRYPT*.

Gennaro, R., Ishai, Y., Kushilevitz, E., and Rabin, T. (2001). The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *STOC*.

Gennaro, R., Jarecki, S., Krawczyk, H., and Rabin, T. (2007). Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*.

Gennaro, R., Rabin, M. O., and Rabin, T. (1998). Simplified vss and fast-track multiparty computations with applications to threshold cryptography. PODC.

gfshare (2010). gfshare. http://manpages.ubuntu.com/manpages/xenial/en/man7/gfshare.7.html. accessed: 2016-12-14.

Happe, A., Krenn, S., and Lorünser, T. (2016). Malicious Clients in Distributed Secret Sharing Based Storage Networks. In *SPW*.

Herranz, J. and Sáez, G. (2003). Verifiable Secret Sharing for General Access Structures, with Application to Fully Distributed Proxy Signatures. In *FC*.

Jhanwar, M. P., Venkateswarlu, A., and Safavi-Naini, R. (2014). Paillier-Based Publicly Verifiable (Non-Interactive) Secret Sharing. *Designs, Codes and Cryptography*.

Kate, A., Zaverucha, G. M., and Goldberg, I. (2010). *Constant-Size Commitments to Polynomials and Their Applications*.

Katz, J., Koo, C., and Kumaresan, R. (2008). Improving the Round Complexity of VSS in Point-to-Point Networks. In *ICALP (2)*.

Kaya, K. and Selçuk, A. A. (2008). A Verifiable Secret Sharing Scheme Based on the Chinese Remainder Theorem. In *INDOCRYPT*.

Krawczyk, H. (1993). Secret Sharing Made Short. In *CRYPTO*.

Kumaresan, R., Patra, A., and Rangan, C. P. (2010). *The Round Complexity of Verifiable Secret Sharing: The Statistical Case*.

Lorünser, T., Happe, A., and Slamanig, D. (2015). ARCHISTAR: Towards Secure and Robust Cloud Based Data Sharing. In *CloudCom '15*.

Naor, M. and Pinkas, B. (2001). *Efficient Trace and Revoke Schemes*. FC.

Patra, A., Choudhary, A., Rabin, T., and Rangan, C. P. (2009a). *The Round Complexity of Verifiable Secret Sharing Revisited*.

Patra, A., Choudhary, A., and Rangan, C. P. (2009b). Simple and Efficient Asynchronous Byzantine Agreement with Optimal Resilience. In *PODC*.

Pedersen, T. P. (1991). Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. CRYPTO.

Rabin, T. and Ben-Or, M. (1989). Verifiable secret sharing and multiparty protocols with honest majority. STOC.

Schoenmakers, B. (1999). A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic. In *CRYPTO*.

Shamir, A. (1979). How to Share a Secret. *Communications of the ACM*.

Stadler, M. (1996). Publicly Verifiable Secret Sharing. In *EUROCRYPT*.

Symantic (2012). Symatec How to: Split and Rejoin PGP Desktop 8.x keys. https://support.symantec.com/en_US/article.HOWTO41916.html. accessed: 2016-12-14.

Welch, L. and Berlekamp, E. (1983). Error Correction of Algebraic Block Codes. US Patent #4,633,470.

Zhang, J. and Zhang, F. (2014). Information-Theoretical Secure Verifiable Secret Sharing with Vector Space Access Structures over Bilinear Groups. In *ISPEC*.