Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography*

Masayuki Abe[1] and Serge Fehr[2]

[1] NTT Laboratories, Japan
abe@isl.ntt.co.jp
[2] ACAC**, Department of Computing, Macquarie University, Australia
sfehr@ics.mq.edu.au

Abstract. We propose the first distributed discrete-log key generation (DLKG) protocol from scratch which is adaptively-secure in the non-erasure model, and at the same time completely avoids the use of interactive zero-knowledge proofs. As a consequence, the protocol can be proven secure in a universally-composable (UC) like framework which prohibits rewinding. We prove the security in what we call the single-inconsistent-player (SIP) UC model, which guarantees arbitrary composition as long as all protocols are executed by the same players. As applications, we propose a fully UC threshold Schnorr signature scheme, a fully UC threshold DSS signature scheme, and a SIP UC threshold Cramer-Shoup cryptosystem.
Our results are based on a new adaptively-secure Feldman VSS scheme. Although adaptive security was already addressed by Feldman in the original paper, the scheme requires secure communication, secure erasure, and either a linear number of rounds or digital signatures to resolve disputes. Our scheme overcomes all of these shortcomings, but on the other hand requires some restriction on the corruption behavior of the adversary, which however disappears in some applications including our new DLKG protocol.
We also propose several new adaptively-secure protocols, which may find other applications, like a distributed trapdoor-key generation protocol for Pedersen's commitment scheme, an adaptively-secure Pedersen VSS scheme (as a *committed* VSS), or distributed-verifier proofs for proving relations among commitments or even any NP relations in general.

1 Introduction

A distributed key generation protocol is an essential component in threshold cryptography. It allows a set of n players to jointly generate a key pair, (pk, sk), that follows the distribution defined by the target cryptosystem, without the need for a trusted party. While the public-key pk is output in clear, the corresponding secret-key sk remains hidden and is maintained in a shared manner among the players via a secret sharing scheme. This should allow the players to later use sk without explicitly having to reconstruct it. The distributed key-generation for discrete-log based schemes, DLKG in short, amounts to the joint generation of a random group element y as public-key and a sharing of its discrete-log (DL) $x = \log_g(y)$ as secret-key with regard to some given base g. A DLKG protocol must remain secure in the presence of a malicious adversary who may corrupt up to a minority of the players and make them behave in an arbitrary way. Informally, it is required that, for any adversary, y must be uniformly distributed, and the adversary must learn nothing about x beyond $y = g^x$.

DLKG was first addressed by Pedersen in [18]. Gennaro *et al.* pointed out that Pedersen's scheme is not secure against a rushing adversary (and even against a non-rushing adversary) and proposed a new (statically) secure scheme [14]. Then Frankel *et al.* and Canetti *et al.*

* This is the full version of [3].
** Centre for Advanced Computing - Algorithms and Cryptography.

introduced in [13] respectively [7] adaptively secure schemes in the erasure model, and Jarecki and Lysyanskaya improved the schemes to work in the non-erasure model and to remain secure under concurrent composition [16].

These DLKG protocols which are secure against an adaptive adversary rely heavily on the use of interactive zero-knowledge proofs. This poses the question whether this is an inherent phenomenon for adaptively secure DLKG. We answer this question in the negative. Concretely, we propose an adaptively-secure distributed key-generation protocol from scratch which completely avoids the use of interactive zero-knowledge proofs. As a consequence, the protocol can be and is proven secure in a relaxed version of Canetti's universally-composable (UC) framework [4], which prohibits rewinding. We show the usefulness of our distributed key-generation protocol by showing how it gives rise to a *fully* UC threshold Schnorr signature scheme as well as a *fully* UC threshold DSS signature scheme. To the best of our knowledge, these are the first threshold schemes proven secure in the UC framework. We also point out how to combine our results with [16] to get a threshold Cramer-Shoup cryptosystem provably secure in the relaxed UC framework.

The relaxed UC framework, which we call the single-inconsistent-player (SIP) UC framework, coincides with the original UC framework, except that the simulator is allowed to fail in case the adversary corrupts some designated player $P_{j^\star}$, which is chosen at random from the set of all players and announced to (and only to) the simulator. This relaxation still allows for a powerful composition theorem in that protocols may be arbitrary composed, as long as all subsidiary protocols involve the same set of players.

We stress once more that this relaxation only applies to the proposed distributed key-generation protocol but *not* to its application for the threshold signature schemes.

Our DLKG protocol is based on a new adaptively-secure version of Feldman's famous (statically secure) VSS scheme. Although adaptive security was already addressed by Feldman in the original paper [12], and besides the well known standard Feldman VSS scheme he also proposed an adaptively-secure version, the proposed scheme has several shortcomings: (1) it requires the players to be able to reliably erase data, (2) it either proceeds over a linear number of rounds or otherwise needs to incorporate signatures as we will point, and (3) it requires secure communication channels (or expensive non-committing encryption schemes). We propose a new variant of Feldman's VSS scheme which overcomes all of these limitations. Even though the proposed scheme is not fully adaptively secure but requires some restriction on the corruption behavior of the adversary, this restriction is acceptable in that it disappears in the above applications to threshold cryptography.

Furthermore, as building blocks for the above schemes or as related constructions, we also propose several adaptively-secure protocols of independent interest which may very well find other applications: a simple modification of Feldman's adaptively-secure VSS scheme which overcomes (1) and (2) above, though not (3), but is *fully* adaptively-secure, a new adaptively-secure distributed trapdoor-key generation protocol for Pedersen's commitment scheme, an adaptively secure version of Pedersen's VSS scheme as a *committed* VSS, and (distributed-verifier) zero-knowledge proofs in the UC model.

The paper is organized as follows. Section 2 reviews the model we are considering. It includes an introduction to the UC framework of Canetti and the new SIP UC framework. In Sect. 3 we recall Feldman's statically and adaptively secure VSS schemes, and we point out an obstacle in the dispute resolution phase of the adaptive scheme, before we construct our version in Sect. 4. Finally, Sect. 5 shows the applications to adaptively-secure DLKG and universally-composable threshold cryptography, and some related constructions are given in Sect. 6. For improved readability, some lengthy proofs have been moved to the appendix.

2 Preliminaries

2.1 Communication Model

We consider a *synchronized authenticated-link model* where communication is divided up into globally clocked *rounds* in that a message sent off by a player P_s at the beginning of a round is guaranteed to be delivered to the recipient P_r within this round, and where a message seemingly originating from P_s is accepted by the receiver P_r if and only if it indeed has been sent by P_s. Moreover, we assume a broadcast channel with which every player can send a message authentically and all players receive that message (within the same round).

In our security model, Canetti's universally composable framework (see Sect. 2.3 below), which per-se only incorporates unauthenticated point-to-point communication, the above assumed channels are modeled by so called functionalities. The authenticated point-to-point communication is modeled by functionality $\mathcal{F}_{\text{AUTH}}$, which on receiving $(\textsf{send}, sid, P_r, m)$ from P_s sends (sid, P_s, P_r, m) to P_r and the adversary; and the broadcast channel is modeled by functionality $\mathcal{F}_{\text{BC}}$, which on receiving $(\textsf{send}, sid, m)$ from P_s sends (sid, P_s, m) to all players and the adversary. For notational simplicity, however, we treat these functionalities as being part of the communication model, and we simply say "P_s sends m to P_r" and "P_s broadcasts m" rather than "P_s sends $(\textsf{send}, sid, P_r, m)$ to $\mathcal{F}_{\text{AUTH}}$" respectively "$P_s$ sends $(\textsf{send}, sid, m)$ to $\mathcal{F}_{\text{BC}}$".

2.2 The Adversary

We consider an adversary $\mathcal{A}$ which may *corrupt* players at will. Corrupting a player P_j allows $\mathcal{A}$ to read P_j's internal state and to act on P_j's behalf from that point on. In the non-erasure model which we consider here, $\mathcal{A}$ additionally gains P_j's complete history. $\mathcal{A}$ is said to corrupt a player P_j *statically*, if $\mathcal{A}$ corrupts P_j *before* the protocol starts, and $\mathcal{A}$ is said to corrupt P_j *adaptively*, if $\mathcal{A}$ corrupts P_j *during* the execution of the protocol, depending on what $\mathcal{A}$ has seen so far. Correspondingly, $\mathcal{A}$ is called *static* or *adaptive*, depending on whether $\mathcal{A}$ corrupts the players statically or adaptively. Per default, we assume $\mathcal{A}$ to be adaptive. Furthermore, we allow $\mathcal{A}$ to be *rushing*. This means that $\mathcal{A}$ can read the messages that are sent in some round by the uncorrupted players before having to decide on the messages for the corrupt players in this round. Finally, $\mathcal{A}$ is called *t-limited* if it corrupts at most t players.

2.3 Canetti's Universally Composable Framework

In order to formally specify and prove the security of our protocols, we will use the universally composable (UC) framework of Canetti [4]. In this framework, a protocol π is compared with an *ideal functionality* $\mathcal{F}$. Such a functionality can be thought of as a trusted party with whom every player can communicate in a secure (meaning private and authentic) way. There is a number of commands specified that $\mathcal{F}$ will execute. Every player can securely send a command to $\mathcal{F}$, and $\mathcal{F}$ will faithfully carry out the command according to its specification, and may send results back securely to (some of) the players and the adversary.

Many cryptographic constructions – including ours – actually aim at building a protocol π for the players only (without a trusted party) that does "the same thing" as some ideal functionality $\mathcal{F}$, even in the presence of an adversary $\mathcal{A}$. The framework provides on one hand a precise definition of what it means that a protocol π securely realizes $\mathcal{F}$. On the other hand it provides the following composition theorem. For any protocol ρ that securely realizes functionality $\mathcal{G}$ in the so-called $\mathcal{F}$-hybrid model, meaning that it may use $\mathcal{F}$ as a subroutine, composed protocol ρ^π that replaces $\mathcal{F}$ with a secure protocol π also securely realizes $\mathcal{G}$ (in the real-life model).

To prove that a protocol π securely realizes $\mathcal{F}$, one has to construct, for every adversary $\mathcal{A}$ attacking the protocol in question, an ideal-life adversary, or *simulator* $\mathcal{S}$, which gets to attack an ideal scenario where only the players and $\mathcal{F}$ are present. The goal of $\mathcal{S}$ is to achieve "the same" as $\mathcal{A}$ could have achieved by an attack on the real protocol. In the framework, this is formalized by considering an *environment* $\mathcal{Z}$ which provides inputs to and collects outputs from the honest players and can communicate in the real-life execution with $\mathcal{A}$ and in the ideal-life execution with $\mathcal{S}$, and it is required that it cannot tell the difference.

Formally, let the random variable $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, a)$ denote the output of $\mathcal{Z}$ with input $a \in \{0,1\}^*$ after observing the real-life computation with security parameter $\kappa \in \mathbb{N}$ and uniformly chosen randomness for every player and $\mathcal{A}$. Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble of $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, a)$ for all κ and a. Similarly, let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, a)$ and $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the output of $\mathcal{Z}$ and its ensemble, respectively, regarding the ideal-model computation. The protocol π is said to securely realize $\mathcal{F}$ (in the real-life model), or simply to be secure, if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$, such that $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ for every environment $\mathcal{Z}$, where $\approx$ denotes computational indistinguishability: for all positive constant $c \in N$ and for all sufficiently large κ and all $a \in \{0,1\}^*$,

$$\big|\Pr[\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, a) = 1] - \Pr[\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, a)]\big| < \kappa^{-c}\,.$$

This extends to the security of protocols in the hybrid model. That is, protocol ρ in the $\mathcal{F}$-hybrid model securely realizes functionality $\mathcal{G}$ if for every hybrid-model adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that $\text{HYB}^{\mathcal{F}}_{\rho,\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}_{\mathcal{G},\mathcal{S},\mathcal{Z}}$ for all environments $\mathcal{Z}$, where $\text{HYB}^{\mathcal{F}}_{\rho,\mathcal{A},\mathcal{Z}}$ is the ensemble of the output of $\mathcal{Z}$ interacting with parties running ρ in the $\mathcal{F}$-hybrid model. Then, the composition theorem guarantees that if protocol ρ securely realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model and π securely realizes $\mathcal{F}$ in the real-life model, then the composed protocol ρ^π securely realizes $\mathcal{G}$ in the real-life model. For more details, see [4].

In proofs of this type of security, the simulator $\mathcal{S}$, which is running in the ideal-life execution with players $\tilde{P}_1, \ldots, \tilde{P}_n$, typically works as follows. $\mathcal{S}$ internally runs a copy of the adversary $\mathcal{A}$ and, by impersonating the real-life players $P_1, \ldots, P_n$ (and functionality $\mathcal{F}$ in the $\mathcal{F}$-hybrid model), $\mathcal{S}$ simulates $\mathcal{A}$'s view of an execution of the protocol consistent with the in- and output(s) in the ideal-life execution. Any interaction between $\mathcal{Z}$ and $\mathcal{A}$ is passed back and forth with no change. If $\mathcal{S}$ can simulate $\mathcal{A}$'s view such that together with the outputs of the honest $\tilde{P}_i$'s it is computationally indistinguishable from $\mathcal{A}$'s view and the honest P_i's outputs in a real execution with the same inputs, then $\mathcal{Z}$ will not be able to tell any difference.

2.4 Single-Inconsistent-Player UC Framework

The single-inconsistent-player (SIP) technique of [7] is often used to achieve both adaptive security and efficiency. A protocol in the SIP model is secure (i.e. securely simulatable in the classical model of computation) if the adversary does not corrupt a designated player which is chosen independently at random before the protocol starts. Using the terms of the UC framework, it means that the simulator $\mathcal{S}$ is given as input the identity of a randomly chosen player $P_{j^\star}$, and $\mathcal{S}$ is required to work well as long as $P_{j^\star}$ is uncorrupted. In the case of t-limited adversary with $t < n/2$, this reduces $\mathcal{S}$'s success probability by a factor of $1/2$. This still guarantees security in that whatever $\mathcal{A}$ can do in the real-life model, $\mathcal{S}$

has a good chance in achieving the same in the ideal-life model. Indeed, in the classical sense, a simulator is considered successful if it works better than with negligible probability. However, with such a simulator $\mathcal{S}$, the composition theorem no longer works in its full generality. To minimize the effect of the SIP approach, we have to limit the set of players to be the same in all subsidiary protocols. This way, $P_{j^\star}$ can be sampled once and for all, and the condition that $P_{j^\star}$ remains uncorrupted applies to (and either holds or does not hold) simultaneously for all protocols. With this limitation, the composition theorem essentially works as before.

This is formalized as follows. We include the choice of $j^\star$ in the probability space and extend the notation $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, a)$ to represent the output distribution of $\mathcal{Z}$ for such a $\mathcal{S}$. Let corr denote the event that $P_{j^\star}$ is corrupted. Suppose that protocol π securely realizes functionality $\mathcal{F}$ in the SIP UC model in the sense that for any adversary $\mathcal{A}$, there exists $\mathcal{S}_\pi$ such that for any environment $\mathcal{Z}$

$$\left| \Pr[\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, a) = 1 | \neg\mathsf{corr}] - \Pr[\text{IDEAL}_{\mathcal{F},\mathcal{S}_\pi,\mathcal{Z}}(\kappa, a) | \neg\mathsf{corr}] \right| < \kappa^{-c},$$

for any constant c and sufficiently large κ. In short, $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}|\neg\mathsf{corr}} \approx \text{IDEAL}_{\mathcal{F},\mathcal{S}_\pi,\mathcal{Z}|\neg\mathsf{corr}}$. Also suppose that a $\mathcal{F}$-hybrid model protocol ρ securely realizes functionality $\mathcal{G}$ in the SIP UC model in that for every $\mathcal{A}$ there exists $\mathcal{S}_\rho$ such that $\text{HYB}^{\mathcal{F}}_{\rho,\mathcal{A},\mathcal{Z}|\neg\mathsf{corr}} \approx \text{IDEAL}_{\mathcal{G},\mathcal{S}_\rho,\mathcal{Z}|\neg\mathsf{corr}}$ for all $\mathcal{Z}$. Then, the SIP universal composition theorem states that the composed protocol ρ^π securely realizes $\mathcal{G}$ in the (real-life) SIP UC model, i.e., for every $\mathcal{A}$ there exists $\mathcal{S}$ such that $\text{REAL}_{\rho^\pi,\mathcal{A},\mathcal{Z}|\neg\mathsf{corr}} \approx \text{IDEAL}_{\mathcal{G},\mathcal{S},\mathcal{Z}|\neg\mathsf{corr}}$ for all $\mathcal{Z}$. This theorem can be proven essentially in the same way as for the original UC theorem with some minor adjustments. $\mathcal{S}$ mainly behaves as $\mathcal{S}_\rho$ does. For every invocation of sub-protocol π within ρ, $\mathcal{S}$ simulates the sub-protocol by invoking $\mathcal{S}_\pi$ with the same $j^\star$ given to $\mathcal{S}$. In this way, all simulation is done with respect to the same SIP $P_{j^\star}$ and corr can be treated as a global condition that holds or fails in all invocation of sub-protocols at the same time. Since $\neg\mathsf{corr}$ happens with probability at least $1/2$ for any $n/2$-limited adversary, $\mathcal{S}$ achieves good quality of reduction for evaluating the security of composed protocol ρ^π.

2.5 Modeling Secure Message Transmission and Committed VSS

Secure Message Transmission (SMT): Following [4], secure communication is modeled by the following functionality.

Definition 1 (Secure Message Transmission Functionality: $\mathcal{F}_{\text{SMT}}$).
On receiving $(\mathsf{send}, sid, P_r, m)$ *from* P_s, $\mathcal{F}_{\text{SMT}}$ *sends* (sid, P_s, m) *to* P_r *and* (sid, P_s, P_r) *to* $\mathcal{S}$.

Note that it is required by definition of the hybrid-model that (sid, P_s, P_r) is sent to $\mathcal{S}$. If the length of m may vary, it is also given to $\mathcal{S}$.

This functionality, however, cannot be realized in the synchronized communication model where messages will never be blocked by the adversary. This is important for us since we assume the use of a broadcast channel which essentially results in assuming all communication be done in a synchronized way. It is known that such $\mathcal{F}_{\text{SMT}}$ can be realized in the authenticated and asynchronous communication model where $\mathcal{S}$ can block the message from $\mathcal{F}_{\text{SMT}}$ to P_r. The realization, say π_{SMT}, requires the sender and the receiver to interact in order to complete the transmission. Now, when $\mathcal{F}_{\text{SMT}}$ is considered in the synchronized communication model where $\mathcal{S}$ cannot block the message from $\mathcal{F}_{\text{SMT}}$ to the receiver, the message transmission in the ideal-process is *atomic* in such a sense that once it is invoked it is always completed. On the other hand, if π_{SMT} is first invoked by an honest sender and the sender is later corrupted by $\mathcal{A}$ and ordered to halt or to lead the receiver to reject, $\mathcal{S}$ cannot simulate such a situation in the ideal-process since the message once sent off from the honest sender has to be delivered to the receiver. So in the current model, π_{SMT} must be non-interactive to securely realize $\mathcal{F}_{\text{SMT}}$ in the synchronized communication model in the presence of an adaptive adversary.[3] Although aborting the protocol is harmless as long as its aftermath can be treated in a correct way and the model should be changed to be able to handle this type of really ideal functionalities, we decided to ease the functionality to fit to the current model so that unexpected flaws caused by the change of the model can be avoided. Concretely, we consider *spooled* message transmission functionality, denoted by $\mathcal{F}_{\text{SSMT}}$, to capture the case where as a consequence of a corruption the sender changes its mind during the execution of the protocol.

Definition 2 (Spooled SMT Functionality: $\mathcal{F}_{\text{SSMT}}$).

1. *On receiving* $(\mathsf{spool}, sid, P_r, m)$ *from* P_s*, record* m *and send* $(\mathsf{spooled}, sid, P_s, P_r)$ *to* $\mathcal{S}$*.*
2. *On receiving* (send, sid) *from* P_s*, send* (sid, P_s, m) *to* P_r *and* $(\mathsf{received}, sid)$ *to* $\mathcal{S}$*; or, on receiving* (send, sid, m') *from* corrupted P_s*, send* (sid, P_s, m') *to* P_r*.*

In the definition of this functionality, we implicitly understand that each step is performed only once and in order. This rule applies to all functionalities defined in this paper, unless explicitly stated otherwise.

Finally, the following extended version of $\mathcal{F}_{\text{SSMT}}$ allows the sender, in case of a dispute, to convince the other players of the message m sent to the receiver.

Definition 3 (Spooled SMT Functionality with Opening: $\mathcal{F}_{\text{SSMTwO}}$).

1. *On receiving* $(\mathsf{spool}, sid, P_r, m)$ *from* P_s*, send* $(\mathsf{spooled}, sid, P_s, P_r)$ *to* $\mathcal{S}$*.*
2. *On receiving* (send, sid) *from* P_s*, send* (sid, P_s, m) *to* P_r *and* $(\mathsf{received}, sid)$ *to* $\mathcal{S}$*; or, on receiving* (send, sid, m') *from* corrupted P_s*, send* (sid, P_s, m') *to* P_r*, and set* $m := m'$*.*
3. *On receiving* (open, sid) *from* P_s*, send* $(\mathsf{sent}, sid, P_s, P_r, m)$ *to all players and* $\mathcal{S}$*.*

It is important to see that (open, sid) is executed only if it is given from P_s. In some sense, the sender commits the message to the network and opens it when needed.

As one can imagined, such a functionality could be realized in the straightforward way in the secure-channel model by using digital signature functionality; the receiver signs the received message and returns the signature to the sender through a secure-channel.

Committed VSS: An advantage of using Feldman and Pedersen VSS in protocol design is that besides producing a (correct) sharing, they also commit the dealer to the shared secret. Often, this commitment can be and is used in upper-level protocols. However, in the definition of UC-secure VSS given in [4], such a commitment is hidden in the protocol and not part of the functionality, and thus not available for external protocols. We introduce the notion of *committed* VSS to overcome this inconvenience.

Let $\mathsf{com}_K : S_K \times R_K \to Y_K$ be a (efficiently computable) commitment function, indexed by a *commitment key* K. Typically, K is sampled by a poly-time generator (on input the security parameter κ). A commitment for a secret $s \in S_K$ is computed as $y = \mathsf{com}_K(s; r)$,

[3] As pointed out by Nielsen [17], a similar problem also appears in the context of commitment schemes: the standard commitment functionality cannot be securely realized by an interactive protocol (in which the first message is not yet committing), *not even in the asynchronous communication model.* This uncovers a flaw in essentially all "proven-secure" UC commitments [6, 11].

where we use the semicolon ';' to express that the second argument, r, is chosen randomly (from R_K) unless it is explicitly given.

A committed VSS is a type of VSS where the dealer is committed to the shared secret s by $\mathsf{com}_K(s;r)$ as a result of the protocol execution. It is understood as a VSS whose sharing phase leaks nothing but $\mathsf{com}_K(s;r)$ with regard to secret s. We formally model this notion for a threshold access structure (with threshold $t+1$) by the following functionality.

Definition 4 (Committed Verifiable Secret Sharing Functionality: $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{com}_K}$).

1. *On receiving* $(\mathsf{share}, sid, (s,r))$ *from dealer* P_d*, send* $(\mathsf{shared}, sid, P_d, \mathsf{com}_K(s;r))$ *to all players and* $\mathcal{S}$*.*
2. *On receiving* (open, sid) *from* $t+1$ *distinct players send* $(\mathsf{opened}, sid, s)$ *to all players and* $\mathcal{S}$*.*

Optionally, $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{com}_K}$ might also be instructed to additionally announce r in step 2. Because of the same reason as in the modeling of the secure message transmission, we need to allow an adaptively corrupted dealer to change his mind during the protocol execution. As above, this is done by incorporating spooling into the VSS functionality.

Definition 5 (Committed Verifiable Secret Sharing with Spooling: $\mathcal{F}_{\mathrm{SVSS}}^{\mathsf{com}_K}$).

1. *On receiving* $(\mathsf{spool}, sid, (s,r))$ *from dealer* P_d*, send* $(\mathsf{spooled}, sid, P_d, y)$ *to* $\mathcal{S}$ *where* $y = \mathsf{com}_K(s;r)$*.*
2. *On receiving* (share, sid) *from* P_d*, send* $(\mathsf{shared}, sid, P_d, y)$ *to all players and* $\mathcal{S}$*; or, on receiving* $(\mathsf{share}, sid, (s',r'))$ *from corrupt* P_d*, send* $(\mathsf{shared}, sid, P_d, y')$ *to all players where* $y' = \mathsf{com}_K(s';r')$ *and set* $s := s'$*.*
3. *On receiving* (open, sid) *from* $t+1$ *distinct players, send* $(\mathsf{opened}, sid, s)$ *to all players and* $\mathcal{S}$*.*

We stress that, in both definitions, y must be determined only from the input. This is because in the real world, y is chosen by the possibly corrupt dealer and may follow a distribution the simulator does not know. Hence it is too much demanding that r is generated inside the ideal functionality.[4]

We would like to mention that for certain candidate protocols π_{VSS} for committed VSS (with spooling), whose security rely on the commitment scheme com_K, the generation of the key K needs to be added to the VSS functionality in order to be able to prove π_{VSS} secure in the UC framework. This is for instance the case for Pedersen's VSS scheme as discussed in Section 6.2.

2.6 The Discrete-Log Setting

Let κ be a security parameter and q be a prime of size κ. Let G_q denote a group of order q, and let g be a generator. We use multiplicative notation for the group operation of G_q. Some of our constructions require G_q to be the order-q multiplicative subgroup of $\mathbb{Z}_p^*$ with prime $p = 2q+1$. Unless otherwise noted, all arithmetics are done in $\mathbb{Z}_q$ or G_q and should in each case be clear from the context.

[4] This observation leads to an interesting consequence, namely that Pedersen VSS is not secure in the UC framework if y is considered as a part of its output despite the fact that the shared secret is perfectly independent of the joint view of any t corrupted players. This subject is discussed more extensively in Section 6.2.

Throughout, we assume that such (G_q, q, g) is given to all players, and that the Decision Diffie-Hellman problem for (G_q, q, g) is intractable, meaning that the respective uniform distributions over $\mathsf{DH} = \{(g^\alpha, g^\beta, g^\gamma) \in {G_q}^3 \mid \alpha \cdot \beta = \gamma\}$ and $\mathsf{RND} = {G_q}^3$ are computationally indistinguishable. This assumption implies the discrete-log assumption for (G_q, q, g): given a random $h = g^\omega$, it is computationally infeasible to compute ω.

3 The Original Feldman VSS

The Basic Scheme: Let $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_q$ be distinct and non-zero. In order to share a (random) secret $s \in \mathbb{Z}_q$, the dealer selects a Shamir sharing polynomial $f(X) = s + a_1 X + \cdots + a_t X^t \in \mathbb{Z}_q[X]$ and sends $s_j = f(\alpha_j)$ privately to P_j. Additionally, he broadcasts $C_0 = g^s$ as well as $C_k = g^{a_k}$ for $k = 1, \ldots, t$. Each player P_j now verifies whether

$$g^{s_j} = \prod_{k=0}^{t} C_k^{\alpha_j^k} . \tag{1}$$

If it does not hold for some j, then player P_j broadcasts an *accusation* against the dealer, who has to respond by broadcasting s_j such that (1) holds. If he fails, then the execution is rejected, while otherwise P_j uses the new s_j as his share. Correct reconstruction is achieved simply by filtering out shares that do not satisfy (1).

This scheme is proved secure against a *static* adversary: Assume that $\mathcal{A}$ corrupts t players $P_{j_1}, \ldots, P_{j_t}$. Given $C_0 = g^s$, the simulator $\mathcal{S}$ simply chooses random shares $s_{j_i} \in \mathbb{Z}_q$ $(i = 1 \ldots t)$ for the corrupted players, and it computes $C_1, \ldots, C_t$ with the right distribution from g^s and $g^{s_{j_1}}, \ldots, g^{s_{j_t}}$'s by applying appropriate Lagrange interpolation coefficients "in the exponent". Informally, this shows that $\mathcal{A}$ learns nothing about s beyond g^s.

This simulation-based proof though fails completely if the adversary may corrupt players *adaptively*, i.e., during or even after the execution of the protocol. The problem is that given $C_0 = g^s$, $\mathcal{S}$ needs to come up with $C_1, \ldots, C_t$ such that if $\mathcal{A}$ corrupts some player P_j at some later point, $\mathcal{S}$ can serve $\mathcal{A}$ with s_j such that (1) is satisfied. However, it is not known how to successfully provide such s_j for any dynamic choice of j without knowing s, unless $\mathcal{A}$ corrupts the dealer to start with.

Adaptive Security with Erasure: Feldman addressed adaptive security by providing a set-up phase where the dealer assigns a *private* X-coordinate $\alpha_j \in \{1, \ldots, n\}$ to every P_j. Additionally, he needs to convince the players of the uniqueness of their α_j. This is done in the following way. Let E be a semantically-secure public-key encryption function, with public-key chosen by the dealer.

1. The dealer computes an encryption $A_j = E(j; r_j)$ (with random r_j) for every $j \in \{1, \ldots, n\}$, and he chooses $\alpha_1, \ldots, \alpha_n$ as a random permutation of $1, \ldots, n$. Then, he broadcasts $A_1, \ldots, A_n$ ordered in such a way that A_j appears in α_j-th position, and he privately sends (α_j, r_j) to P_j.
2. Each P_j locates A_j in position α_j and verifies whether $A_j = E(j; r_j)$ and, if it holds, erases r_j. The dealer erases $r_1, \ldots, r_n$, too.

After the erasure is completed, the dealer performs the basic Feldman VSS with X-coordinates $\alpha_1, \ldots, \alpha_n$. We stress that it is important that the erasures of the r_j's must be done before entering to the sharing phase. On reconstruction, each player broadcasts (α_j, s_j).

Since each A_j can be opened only to j, player P_j is convinced of the uniqueness of α_j. Simulation against an adaptive adversary is argued separately for each phase. If a player gets corrupted in the set-up phase, the simulator $\mathcal{S}$ just honestly gives the internal state of the corrupt player to the adversary. Nothing needs to be simulated. Then, the sharing phase is simulated similar as for the static adversary, except that, since $\mathcal{S}$ does not know which players will be corrupted, it predetermines shares for a *random* subset of size t of the X-coordinates $\{1, \ldots, n\}$, and whenever a player P_j gets corrupted one of these prepared X-coordinates is assigned to P_j as his α_j. Since r_j has already been erased, it is computationally infeasible to determine whether A_i in position α_j is an encryption of j or not.

An Obstacle in Dispute Resolution: We identify a problem in the dispute resolution of the above scheme.[5] Suppose that honest P_j accuses the dealer, and that instead of publishing correct (α_j, s_j), the corrupt dealer responds by publishing (α_i, s_i) of another honest player P_i. Since r_j and r_i have been already erased, both P_j and P_i have no means to prove that the published α_i is different from the original assignment.

To efficiently settle such a dispute, digital signatures are needed: the dealer sends α_j together with his signature in the set-up phase. This allows P_j to publish the signature when he accuses the dealer in the sharing phase. Without using digital signatures, $O(t)$ additional rounds are needed to settle the dispute: If P_i observes that his (α_i, s_i) is published to respond to the accusation from P_i, P_i also accuses the dealer and the dealer publishes the data for P_i this time. After repeating this accuse-then-publish process at most $t+1$ times, the dealer either gets stuck or exposes $t+1$ correct shares.

4 Adaptive Security without Overheads and Erasures

The goal of this section is an adaptively secure Feldman VSS that provides (1) security without the need for reliably erasing data, (2) efficient dispute resolution without digital signatures, and (3) efficient realization over a public network, i.e. without secure channels (or expensive non-committing encryptions).

The first two goals are achieved by a simple modification of the original Feldman VSS. The idea is to replace the encryption function E with instantiations of a trapdoor commitment scheme with certain properties whose commitment keys are provided separately from each player so that the trapdoors are not known to the dealer. We show this modified Feldman VSS and the security proof in Sect. 6.1. Since Pedersen's commitment scheme turns out to be good enough for this purpose, we have a scheme that meets (1) and (2) solely under the DL assumption. Furthermore, the modified scheme is more efficient in the number of communication rounds over the original adaptively-secure Feldman VSS.

Hence, what the secure-channels model is concerned, we are done. Unfortunately, we do not know how to efficiently implement the above scheme efficiently over a public network, even when limiting the power of the adversary as we do in Sect. 4.2 below. Therefore, we design a new scheme which allows to seamlessly install our efficient components for public communication presented later.

[5] No dispute resolution procedure is shown in [12]. It is said that a player simply rejects the dealer when he receives an incorrect share (and the dealer is disqualified if more than $t+1$ players rejects). But this works only if $t < n/3$.

4.1 Construction in a Hybrid Model

Our approach is to let each player P_j select a random non-zero X-coordinate $\alpha_j \in \mathbb{Z}_q$ and send it privately to the dealer. When corrupted, a simulated player reveals a (fake) X-coordinate that has been prepared in advance to be consistent with the transcript, as in Feldman's approach. On the other hand, in case of a dispute, each player P_j should be able to convince the other players of his α_j. This is achieved by initially sending α_j to the dealer using secure message transmission *with opening*, as specified in Sect. 2.5 by functionality $\mathcal{F}_{\text{SSMTwO}}$. The scheme is detailed in Fig. 1 in the ($\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{F}_{\text{SSMT}}$)-hybrid model.

[Sharing Phase]

F-1. Each P_j selects $\alpha_j \leftarrow \mathbb{Z}_q^*$ and sends α_j to the dealer via $\mathcal{F}_{\text{SSMTwO}}$. The dealer replaces any α_j that happens to be 0 by $\alpha_j = 1$.

F-2. The dealer selects $f(X) = a_0 + a_1X + \cdots + a_tX^t \leftarrow \mathbb{Z}_q[X]$ where $a_0 = s$ and computes $C_k = g^{a_k}$ for $k = 0, \ldots, t$ and broadcasts $(C_0, \ldots, C_t)$. To every P_i he sends $s_i = f(\alpha_i)$ by using $\mathcal{F}_{\text{SSMT}}$.

F-3. Each P_j verifies $g^{s_j} = \prod_{k=0}^{t} C_k^{\alpha_j^k}$ and broadcasts *verified* if it holds. Otherwise, P_j broadcasts *accuse dealer*. For every accusation, the following sub-protocol is executed in parallel.

  (a) P_j sends (*open*, *sid*) to $\mathcal{F}_{\text{SSMTwO}}$ and every player receives α_j. If $\alpha_j = 0$, then it is replaced by $\alpha_j = 1$.

  (b) The dealer broadcasts the corresponding s_j.

  (c) If (α_j, s_j) satisfies the verification predicate, then P_j accepts (α_j, s_j) as his share, otherwise the players output a default sharing of $s = 0$. (Note that the players have agreement on the published values α_j and s_j).

[Reconstruction Phase]

Each P_j broadcasts (α_j, s_j), identifies $\mathcal{Q} \subseteq \{1, \ldots, n\}$, $|\mathcal{Q}| \geq t+1$, so that (α_i, s_i) satisfies the verification predicate for all $i \in \mathcal{Q}$, reconstructs secret s by Lagrange interpolation with regard to $\mathcal{Q}$, and then outputs s.

Fig. 1. Adaptively secure Feldman-VSS π_{XFVSS} in ($\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{F}_{\text{SSMT}}$)-hybrid model.

Consider Feldman's commitment scheme fcom_g with base g: a commitment for a secret $s \in \mathbb{Z}_q$ is computed as $\mathsf{fcom}_g(s; r) = \mathsf{fcom}_g(s) = g^s$ (without using r).

Proposition 1. *Protocol π_{XFVSS} shown in Fig. 1 securely realizes functionality $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$ in the ($\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{F}_{\text{SSMT}}$)-hybrid model against t-limited adaptive adversary for $t < n/2$.*

The proof is given in Appendix A. Essentially, it uses the same idea as Feldman's version: the simulator prepares (random) shares $\tilde{s}_1, \ldots, \tilde{s}_t$ for t X-coordinates $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_t$ and assigns to every newly corrupt player P_j one of these X-coordinates as α_j and the corresponding share as s_j.

4.2 Efficient Composition to the Real-life Protocol

This section provides protocols that realize $\mathcal{F}_{\text{SSMT}}$ and $\mathcal{F}_{\text{SSMTwO}}$ over the public network with broadcast, i.e., without secure channels. Then, by applying the composition theorem, one can have adaptively secure Feldman VSS as a real-life protocol. As we shall see, these realizations are efficient but have some limitation on the adversary, which though can be successfully overcome in our applications.

Our constructions require an efficient bidirectional mapping between $\mathbb{Z}_q$ and G_q while the DDH problem should be hard to solve. This is the case when G_q is the order-q multiplicative subgroup of $\mathbb{Z}_p^*$ with prime $p = 2q+1$. Indeed, encoding $\mathbb{Z}_q \to G_q$ can be done by $m \mapsto M = m^2 \bmod p$, where $m \in \mathbb{Z}_q$ is identified with its representation in $\{1, \ldots, q\}$. This encoder is denoted by $M = \mathit{Encode}(m)$ and the corresponding decoder by $m = \mathit{Decode}(M)$.

Receiver Non-committing Message Transmission: By π_{RNC}, we denote a protocol that realizes $\mathcal{F}_{\mathrm{SMT}}$ (or $\mathcal{F}_{\mathrm{SSMT}}$) with receiver non-committing feature. That is, remains secure even if the receiver is adaptively corrupted (in the non-erasure model), while the sender may only be statically corrupted. Note that with such a restriction on the sender, $\mathcal{F}_{\mathrm{SMT}}$ can be realized (without spooling). We review the construction by [16] (adapted to accept messages $m \in \mathbb{Z}_q$), which is originally designed in a classical model but can fit to the UC model. A proof is given in Appendix B.

A-0.	(Initial step) Sender P_s chooses $h \leftarrow G_q$ and sends it to receiver P_r.
A-1.	P_r selects $z_1, z_2 \leftarrow \mathbb{Z}_q$, computes $y = g^{z_1} h^{z_2}$, and sends y to P_s.
A-2.	P_s computes $u = g^r$, $v = h^r$ and $c = \mathit{Encode}(m)\, y^r$, where $r \leftarrow \mathbb{Z}_q$, and sends (u, v, c) to P_r
A-3.	P_r computes $m = \mathit{Decode}(cu^{-z_1} v^{-z_2})$.

Fig. 2. Protocol π_{RNC} for receiver non-committing transmission.

Lemma 1. *Under the DDH assumption, protocol π_{RNC} securely realizes $\mathcal{F}_{\mathrm{SMT}}$ (or $\mathcal{F}_{\mathrm{SSMT}}$) against an adaptive adversary if the sender is only statically corrupt and $\mathcal{S}$ is aware of ϖ with $\mathit{Encode}(m) = g^{\varpi}$.*

The assumption that the ideal-life adversary $\mathcal{S}$ is aware of the DL of $\mathit{Encode}(m)$ seems quite restrictive for π_{RNC} to be a general stand-alone tool. It is however acceptable for our purpose as m will be chosen by $\mathcal{S}$ in an upper-level protocol (playing the role of the to-be-corrupted sender) such that it knows the DL of $\mathit{Encode}(m)$. We stress that this assumption does not mean at all that $\mathcal{S}$ is given any kind of power to solve the DL problem.

Sender Non-committing Message Transmission with Opening: A protocol π_{SNC} that realizes $\mathcal{F}_{\mathrm{SSMT}}$ with sender non-committing feature follows easily from π_{RNC}. The receiver P_r simply uses π_{RNC} to securely send a randomly chosen $k \in G_q$ to the sender P_s (precisely, P_r sends the message $\mathit{Decode}(k) \in \mathbb{Z}_q$), and then P_s sends $e = k\mathit{Encode}(m)$ to P_r, who computes m as $m = \mathit{Decode}(ek^{-1})$. We also consider the following variant of π_{SNC}, which we denote by π_{SNCwO}. All communication is done over the broadcast channel, and in an additional phase, the *opening phase*, the sender P_s publishes z_1 and z_2, privately sampled for the secure transmission of m, and every player verifies whether $g^{z_1} h^{z_2} = y$ and computes $k = cu^{-z_1} v^{-z_2}$ and $m = \mathit{Decode}(ek^{-1})$. A full description is given in Figure 11 in Appendix C.

Lemma 2. *Under the DDH assumption, protocol π_{SNC} securely realizes $\mathcal{F}_{\mathrm{SSMT}}$ and π_{SNCwO} securely realizes $\mathcal{F}_{\mathrm{SSMTwO}}$ against an adaptive adversary if the receiver is only statically corrupt and $\mathcal{S}$ is aware of ϖ with $\mathit{Encode}(m) = g^{\varpi}$.*

The proof of Lemma 2 is similar to that of Lemma 1, although slightly more involved. For completeness, it is given in Appendix C.

Composition with the Efficient Realizations: We now show that when the functionalities $\mathcal{F}_{\text{SSMTwO}}$ and $\mathcal{F}_{\text{SSMT}}$ in the hybrid-protocol π_{XFVSS} are implemented by π_{SNCwO} and π_{RNC}, respectively, then the composed protocol securely realizes $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ (or $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$) in some weakened sense as stated below.

Theorem 1. *Implementing the functionality $\mathcal{F}_{\text{SSMTwO}}$ in step F-1 of the hybrid-protocol π_{XFVSS} from Fig. 1. by π_{SNCwO} and $\mathcal{F}_{\text{SSMT}}$ in step F-2 by π_{RNC} results in a secure realization of $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ (or $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$) in the real-life model, assumed that (1) the adversary corrupts the dealer only statically, and (2) the adversary corrupts players only before the reconstruction phase.*

Proof. The claim follows essentially from Proposition 1, Lemma 1 and 2, and the composition theorem. It remains to show that the assumptions for Lemma 1 and 2 are satisfied. By assumption (1) it is guaranteed that the receiver in π_{SNC} and the sender in π_{RNC} (which in both cases is the dealer) is only statically corrupt. Furthermore, by (2) and the way $\mathcal{S}$ works in the proof of Proposition 1, the messages, which are supposedly send through $\mathcal{F}_{\text{SSMTwO}}$ and $\mathcal{F}_{\text{SSMT}}$ and for which $\mathcal{S}$ has to convince $\mathcal{A}$ as being the messages sent through $\mathcal{F}_{\text{SSMTwO}}$ respectively $\mathcal{F}_{\text{SSMT}}$ are the values $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_t$ and $\tilde{s}_1, \ldots, \tilde{s}_t$, all chosen randomly from $\mathbb{Z}_q$ (respectively $\mathbb{Z}_q^*$) by $\mathcal{S}$. Hence, $\mathcal{S}$ could sample them just as well by choosing $\varpi \leftarrow \mathbb{Z}_q$ and computing $Decode(g^{\varpi})$ such that the conditions for Lemma 1 and 2 are indeed satisfied. Finally, as the dealer may only be statically corrupted, we do not need to care about spooling. Thus $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ and $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$ are equivalent here. □

5 Applications to Threshold Cryptography

In this section, we propose several applications of the adaptively-secure Feldman VSS scheme from the previous section. Our main applications are a distributed DL-key generation protocol and UC threshold versions of the Schnorr and the DSS signature scheme, though we also propose some related applications which might be of independent interest like a trapdoor-key generation protocol for Pedersen's commitment scheme, and in Sect. 6 a common-random-string generator with applications to zero-knowledge in the UC model and distributed-verifier UC proofs of knowledge. Interestingly, even though our Feldman VSS scheme has restricted adaptive security, the applications remain fully adaptively secure in the (SIP) UC model and do not underly restrictions as posed in Theorem 1.

To simplify terminology, from now on when referring to protocol π_{XFVSS}, we mean π_{XFVSS} from Fig. 1 with $\mathcal{F}_{\text{SSMTwO}}$ and $\mathcal{F}_{\text{SSMT}}$ replaced by π_{SNCwO} and π_{RNC} as specified in Theorem 1. Furthermore, it will at some point be convenient to use a different basis, say h, rather than the public parameter g in the core part of π_{XFVSS}, such that for instance h^s will be published as C. This will be denoted by $\pi_{\text{XFVSS}}[h]$, and obviously securely realizes $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_h}$. We stress that this modification is not meant to affect the sub-protocols π_{SNCwO} and π_{RNC}.

5.1 How to Generate The First Trapdoor Commitment-Key

In many protocols, a trapdoor commitment-key is considered as given by some trusted party so that the trapdoor information is unknown to any player. In order to achieve security *from scratch*, the (trapdoor) commitment-key needs to be generated securely by the players, without a trusted party, and without the use of a common trapdoor commitment-key. In this section, we show such a protocol for Pedersen's commitment scheme.

The protocol, π_{HGEN}, is illustrated in Fig. 3. We assume that it is triggered by a player P_i who sends *init* to all players. The protocol outputs a (trapdoor) commitment-key $h \in G_q$ for Pedersen's commitment scheme. Note that the corresponding trapdoor $\log_g h = \sum_{j\in\mathcal{Q}} \chi_j$ is not shared among the players (in the usual way).

H-1. Every P_j chooses $\chi_j \leftarrow \mathbb{Z}_q$ and sends $(\textit{share}, sid, \chi_j)$ to $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$. Let $\mathcal{Q}$ be the set of players whose $(\textit{shared}, sid, P_j, Y_j)$ is published by $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$. Remember that $Y_j = \mathsf{fcom}_g(\chi_j) = g^{\chi_j}$.
H-2. Every player outputs $h = \prod_{j\in\mathcal{Q}} Y_j$.

Fig. 3. Commitment-key generation protocol π_{HGEN} in $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$-hybrid model.

Unfortunately, one cannot expect h to be random as a rushing party can affect its distribution. However, the protocol inherits the following two properties which are sufficient for our purpose. (1) A simulator that simulates π_{HGEN} can compute the DL of h, and (2) given $Y \in G_q$, a simulator can embed Y into h so that given $\log_g h$, the simulator can compute $\log_g Y$. The latter in particular implies that the adversary is not able to compute the trapdoor $\log_g h$.

Our idea for formally capturing such a notion is that the ideal functionality challenges the adversary $\mathcal{S}$ by sending a random $h' \in G_q$ and allows $\mathcal{S}$ to randomize it so that h' is transformed into h such that $\mathcal{S}$ knows the trapdoor for h if and only if it knows it for h'. This clearly captures (1) and (2) above.

Definition 6 (Commitment-Key Generation Functionality: $\mathcal{F}_{\text{HGEN}}$).

1. *On receiving* $(\textit{generate}, sid)$ *from* P_i*, choose* $h' \leftarrow G_q$ *and send* (h', P_i) *to* $\mathcal{S}$*.*
2. *On receiving* $\gamma \in \mathbb{Z}_q$ *from* $\mathcal{S}$*, compute* $h = h'g^{\gamma}$ *and send* $(\textit{com-key}, sid, h)$ *to all players and* $\mathcal{S}$*.*

Proposition 2. *Protocol* π_{HGEN} *in Fig. 3 securely realizes* $\mathcal{F}_{\text{HGEN}}$ *against* t*-limited adaptive adversary for* $t < n/2$ *in the* $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$*-hybrid SIP UC model.*

Proof. If $\mathcal{A}$ corrupts a player, P_i, before starting π_{HGEN}, $\mathcal{S}$ corrupts $\tilde{P}_i$ and gives its internal state to $\mathcal{A}$. If corrupted P_i initiates π_{HGEN}, $\mathcal{S}$ lets $\tilde{P}_i$ do so by sending $(\textit{generate}, sid)$ to $\mathcal{F}_{\text{HGEN}}$.

On receiving (h', P_i) from $\mathcal{F}_{\text{HGEN}}$, $\mathcal{S}$ lets honest P_i initiate the protocol. (If P_i has been already corrupted, $\mathcal{S}$ can skip this step because the initial message should have been already broadcast by $\mathcal{A}$.) To simulate the SIP $P_{j^\star}$, $\mathcal{S}$ simulates $P_{j^\star}$'s invocation of the sharing phase of $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ with output $(\textit{shared}, sid, h', P_{j^\star})$ as if it has received corresponding input from $P_{j^\star}$. $\mathcal{S}$ also simulates all honest players and their invocation of $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ as prescribed. $\mathcal{S}$ obtains χ_j for all $j \in \mathcal{Q} \setminus \{j^\star\}$ as inputs to $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$. $\mathcal{S}$ then sends $\gamma = \sum_{j\in\mathcal{Q}\setminus\{j^\star\}} \chi_j$ to $\mathcal{F}_{\text{HGEN}}$. This results in receiving $(\textit{com-key}, sid, h)$ from $\mathcal{F}_{\text{HGEN}}$ where $h = h'g^{\gamma}$. $\mathcal{S}$ delivers the message to all players.

It is straightforward to verify that this is a perfect simulation: the only difference to the real-life execution is that $P_{j^\star}$ does not provide his input for $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$, which though is invisible to $\mathcal{A}$ and $\mathcal{Z}$. $\square$

We claim that $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ in π_{HGEN} can be securely realized by the protocol π_{XFVSS} from Theorem 1. This may look contradictory since π_{XFVSS} is secure only against static corruption of the dealer as stated in Theorem 1, while in π_{HGEN} every player acts as a dealer and may be adaptively corrupted. However, looking at the proof, except for the run launched by the SIP $P_{j^\star}$, $\mathcal{S}$ simulates all runs of $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ honestly with true inputs. Hence, for these simulations, the situation is exactly as in the case where the dealer is statically corrupted and the secret is known to the simulator at the beginning. Furthermore, the reconstruction phase of $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ is never invoked in π_{HGEN}. Thus, the following holds.

Theorem 2. *Implementing $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ in π_{HGEN} of Fig. 3 by π_{XFVSS} results in a secure realization of $\mathcal{F}_{\mathrm{HGEN}}$ against t-limited adaptive adversary for $t < n/2$ in the (real-life) SIP UC model.*

5.2 DL-Key Generation

This section constructs an adaptively secure protocol for DLKG, whose functionality is defined below. Clearly, from such a key-generation protocol (respectively functionality), one expects that it outputs a public-key y and in some hidden way produces the corresponding secret-key x (typically by having it shared among the players), such that x can be used to do some cryptographic task like signing or decrypting if enough of the players agree [21]. However, as we want to view our protocol as a generic building block for threshold schemes, we simply require that the secret-key x can be opened rather than be used for some specific task. In Sect. 5.3 and 5.5 we then show concrete example threshold schemes based on our DLKG protocol.

Definition 7 (Threshold DL Key Generation Functionality: $\mathcal{F}_{\mathrm{DLKG}}$).

1. *On receiving $(\mathsf{generate}, sid)$ from P_i, select $x \leftarrow \mathbb{Z}_q$, compute $y = g^x$, and send (key, sid, y) to all players and $\mathcal{S}$.*
2. *On receiving (open, sid) from $t+1$ players, send $(\mathsf{private}, sid, x)$ to all players and $\mathcal{S}$.*

Our realization of $\mathcal{F}_{\mathrm{DLKG}}$ is illustrated in Fig. 5 below, and makes use of (ordinary) Pedersen's VSS scheme given in Fig. 4. We do not prove Pedersen's VSS secure in the UC framework, and in fact it is not (as a committed VSS against an adaptive adversary). The only security requirement we need is covered by the following well-known fact.

Lemma 3. *Except with negligible probability, after the sharing phase of Pedersen's VSS, both the s_i's and r_i's of the uncorrupted players are correct sharings of s and r such that $g^s h^r = C$ and such that s is reconstructed in the reconstruction phase (and s and r coincide with the dealer's choice in case he remains honest), or otherwise $\log_g h$ can be efficiently extracted from the adversary.*

We write $\mathsf{PedVSS}_{g,h}^{j}(s) \rightarrow (s_1, \ldots, s_n, r, C, E_1, \ldots, E_n)$ to denote an execution of the sharing phase of Pedersen's VSS with secret s and player P_j acting as dealer, and with values $s_1, \ldots, s_n, r, C, E_1, \ldots, E_n$ generated as described in Fig. 4.

Note that in π_{DLKG} the *additive* shares x_j are used to reconstruct the secret-key x, rather than the threshold-shares implicitly given by $\xi_j = \sum_i x_{ij}$. The reason is that even though using the threshold shares can be proven secure in the hybrid-model, it resists a security proof when the ideal functionality $\mathcal{F}_{\mathrm{SSMT}}$ in Pedersen's VSS is replaced by π_{RNC} as we do (due to the DL condition from Lemma 1). In Sect. 5.4 we show how to modify the scheme in order to be able to use the threshold-shares as secret-key shares. Also note that using the terminology introduced in [2], based on the results in [1], step K-3 can be seen as a *distributed-verifier zero-knowledge proof* of knowledge of x_j and r_j such that $g^{x_j} = C'_j$ and $h^{r_j} = C''_j$.

[Sharing Phase]

P-1. The dealer selects $f(X) = a_0 + a_1X + \cdots + a_tX^t \leftarrow \mathbb{Z}_q[X]$ and $f'(X) = b_0 + b_1X + \cdots + b_tX^t \leftarrow \mathbb{Z}_q[X]$ where $a_0 = s$. Let $r = b_0$. The dealer then computes and broadcasts $C = C_0 = g^s h^r$ and $C_k = g^{a_k} h^{b_k}$ for $k = 1, \ldots, t$, and he sends $s_i = f(i)$ and $r_i = f'(i)$ to P_i using $\mathcal{F}_{\mathrm{SSMT}}$.

P-2. Each P_i verifies whether $g^{s_i} h^{r_i} = E_i$ where $E_i = \prod_{k=0}^{t} C_k^{i^k}$. P_i broadcasts *verified* if it holds and else initiates the accusation sub-protocol which is the same as that of Feldman VSS with obvious modification.

[Reconstruction Phase]

Every player P_i publicly opens E_i to s_i. The secret s is reconstructed using Lagrange interpolation from the correctly opened s_i's.

Fig. 4. Pedersen's VSS scheme: $\mathsf{PedVSS}_{g,h}(s) \rightarrow (s_1, \ldots, s_n, r, C, E_1, \ldots, E_n)$

[Key-Generation Phase]

K-1. A player, P_i, sends (*generate*, sid) to $\mathcal{F}_{\mathrm{HGEN}}$, and commitment-key h is obtained.

K-2. Each player P_j chooses $x_j \leftarrow \mathbb{Z}_q$ and executes the sharing phase of Pedersen's VSS with secret x_j and commitment-key h: $\mathsf{PedVSS}^{j}_{g,h}(x_j) \rightarrow (x_{j1}, \ldots, x_{jn}, r_j, C_j, E_{j1}, \ldots, E_{jn})$. If a player P_j refuses then a default Pedersen sharing of $x_j = 0$ is taken instead.

K-3. Each P_j sends (*share*, sid_j, x_j) to $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ and (*share*, sid'_j, r_j) to $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_h}$.

K-4. If P_i receives (*shared*, sid_j, P_j, C'_j) and (*shared*, sid'_j, P_j, C''_j), he verifies that $C_j = C'_j C''_j$ holds. (Note that $C'_j = g^{x_j}$ and $C''_j = h^{r_j}$.) If either of such messages has not been received or the relation does not hold, then x_j is reconstructed from its Pedersen sharing, and every P_i sets $C'_j = g^{x_j}$. Output of this phase is the public-key $y = \prod_{j=1}^{n} C'_j$, while each P_j stores x_j as his (additive) secret-key share, to which he is committed by C_j.

[Opening Phase]

Every player P_j publicly opens C_j by broadcasting x_j and r_j. If a player P_j fails to do so, x_j is reconstructed from its Pedersen sharing. Secret-key x is then computed as $x = \sum_{j=1}^{n} x_j$.

Fig. 5. Threshold DLKG protocol π_{DLKG} in $(\mathcal{F}_{\mathrm{HGEN}}, \mathcal{F}_{\mathrm{SSMT}}, \mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g} \mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_h})$-hybrid model.

Theorem 3. *Implementing in the DLKG protocol π_{DLKG} from Fig. 5 the functionalities $\mathcal{F}_{\mathrm{HGEN}}$, $\mathcal{F}_{\mathrm{SSMT}}$, $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ and $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_h}$ by π_{HGEN}, π_{RNC}, $\pi_{\mathrm{XFVSS}}[g]$ and $\pi_{\mathrm{XFVSS}}[h]$, respectively, results in a secure realization of $\mathcal{F}_{\mathrm{DLKG}}$ against adaptive t-limited adversary for $t < n/2$ in the SIP UC model.*

Using the UC with joint state framework [8], one can prove using similar arguments that the commitment-key h can be generated once and for all invocations of π_{DLKG}. Furthermore, concerning efficiency, the communication complexity of the key-generation phase is comparable to that of the schemes by [16]: it requires $O(n^2\kappa)$ bits to be sent over the bilateral public channels and another $O(n^2\kappa)$ bits to be broadcast.

The full proof of Theorem 3 is given in Appendix D. We simply sketch its idea here. First, the simulator $\mathcal{S}$ simulates the generation of h such that it knows the DL of h, while step K-2 is executed as prescribed. Then, it reconstructs the x_i's of the corrupt players, and it computes $C'_{j^\star}$ and $C''_{j^\star}$ for the SIP $P_{j^\star}$ such that $C'_{j^\star} \cdot \prod_{j \neq j^\star} g^{x_j} = y$ and $C_{j^\star} = C'_{j^\star} C''_{j^\star}$, where y is the value provided by $\mathcal{F}_{\mathrm{DLKG}}$. Then it simulates the two Feldman VSSes with $P_{j^\star}$ as dealer, while the other executions are followed as prescribed (with inputs x_j respectively r_j). As a result, the output of the key-generation phase is y. In the opening phase, having received $x = \log_g(y)$ from $\mathcal{F}_{\mathrm{DLKG}}$, $\mathcal{S}$ simply adapts $P_{j^\star}$'s initial $x_{j^\star}$ such that $\sum_j x_j = x$, and it uses the DL of h to open $C_{j^\star}$ to (the new) $x_{j^\star}$. The only difference in the adversary's and thus the environment's view between the simulation and a real execution lies in the encrypted Pedersen shares of (the initial) $x_{j^\star}$ given to the uncorrupted players. By the property of π_{RNC}, this cannot be distinguished by the environment.

From now on, when referring to protocol π_{DLKG}, we mean π_{DLKG} from Fig. 5 with the functionalities replaces by real-life protocols as specified in Theorem 3.

5.3 Universally-Composable Threshold Schnorr-Signatures

As a first example application of our DL-key generation protocol, we propose a threshold variant of Schnorr's signature scheme [19], provable secure in the UC framework.

The Original Scheme: A Schnorr signature for message m under public-key $y = g^x$ consists of (c, s) such that $r = g^s/y^c$ satisfies $H(m, r) = c$, where H is a cryptographic hash-function. Such a signature is computed by the signer (who knows the secret-key x) by choosing $k \leftarrow \mathbb{Z}_q$ and computing

$$r = g^k, \quad c = H(m, r) \quad \text{and} \quad s = k + cx\,.$$

Schnorr's signature scheme can be proven secure, in the sense of existential unforgability against chosen message attacks, in the *random oracle* model. For our threshold DSS signature scheme it is important to note that it does not harm the security of the standard Schnorr signature scheme when r is made available (as it can anyway be computed).

The UC Threshold Scheme: Our threshold version of Schnorr's signature scheme is illustrated in Fig. 6.

[Key-Generation Phase]

The players execute the key-generation phase of π_{DLKG}, resulting in a public-key y, private (additive) secret-key shares $x_1, \ldots, x_n$ with corresponding commitments $C_1, \ldots, C_n$, and commitment-key h.

[Signing Phase]

In order to sign a message m, the following steps are executed.

S-1. The players once more invoke the key-generation phase of π_{DLKG}, but skipping the generation of h and taking h from the generation of y. Denote the output by r, the corresponding additive secret-key shares by $k_1, \ldots, k_n$, and the corresponding commitments by $K_1, \ldots, K_n$.

S-2. Each player P_j computes $c = H(m, r)$ and publicly opens $K_j C_j^c$ to $s_j = k_j + cx_j$. If a player P_j fails to do so, s_j is reconstructed from its Pedersen sharing (which is implicitly given by the Pedersen sharings of x_j and k_j). Signature (c, s) is completed by $s = \sum_j s_j$.

Fig. 6. Threshold Schnorr-signature scheme $\pi_{\text{TSIG}}^{\text{SCHNORR}}$

Consider the ideal threshold signature functionality $\mathcal{F}_{\text{TSIG}}$ by adapting the (single-signer) signature functionality $\mathcal{F}_{\text{SIG}}$ from [5] in the obvious way, as illustrated in Definition 8 below.

Definition 8 (Threshold Signature Functionality: $\mathcal{F}_{\text{TSIG}}$).

1. *On receiving* $(\mathsf{generate}, sid)$ *from* $t{+}1$ *players, hand* (key, sid) *to* $\mathcal{S}$*. On receiving* (key, sid, pk) *from* $\mathcal{S}$ *send* (key, sid, pk) *to all players.*
2. *On receiving* (sign, sid, m) *from* $t + 1$ *players, hand* (sign, sid, m) *to* $\mathcal{S}$*. On receiving* $(\mathsf{signature}, sid, m, \sigma)$ *from* $\mathcal{S}$ *send* $(\mathsf{signature}, sid, m, \sigma)$ *to all players and record* (m, σ)*.*

3. *On receiving* $(\textsf{verify}, sid, m, \sigma, pk')$ *from some player* P_j*, send* $(\textsf{verified}, sid, m, v)$ *to* P_j *and* $\mathcal{S}$*, where* v *is determined as follows. If* $pk = pk'$ *and the pair* (m, σ) *is recorded, then* $v = 1$*. If* $pk = pk'$ *and no pair* (m, σ') *for any* σ' *is recorded, then* $v = 0$*. Otherwise (i.e. if* $pk \neq pk'$ *or if* (m, σ') *is recorded for* $\sigma' \neq \sigma$*) let* $\mathcal{S}$ *decide on the value of* v*.*

After the execution of the first step, the second and third step may be performed an unbounded number of times, and in an arbitrary order.

Theorem 4. *Protocol* $\pi_{\text{TSIG}}^{\text{SCHNORR}}$ *securely realizes* $\mathcal{F}_{\text{TSIG}}$ *against adaptive* t*-limited adversary for* $t < n/2$ *in the UC model, under the DDH assumption and under the assumption that the standard Schnorr signature scheme is secure.*

We stress that interestingly $\pi_{\text{TSIG}}^{\text{SCHNORR}}$ securely realizes $\mathcal{F}_{\text{TSIG}}$ in the *standard* rather than the SIP UC model.

Proof. (Sketch) The simulator $\mathcal{S}$ simply executes *honestly* $\pi_{\text{TSIG}}^{\text{SCHNORR}}$. Note that the public-key y is not dictated by $\mathcal{F}_{\text{TSIG}}$, but rather $\mathcal{F}_{\text{TSIG}}$ asks $\mathcal{S}$ to provide it. Accordingly, $\mathcal{S}$ knows the private key x and can use it in the signing phase of the real-life protocol. In order to prove that this is a good simulation, we argue as follows. The only way $\mathcal{Z}$ may see a difference is when $\mathcal{A}$ breaks the signature scheme, i.e., when a player provides at some point a valid signature on a message that has not been signed. However, if there exist $\mathcal{Z}$ and $\mathcal{A}$ that can enforce such an event with non-negligible probability, then there exists a forger F that breaks the existential unforgability against chosen message attacks of the standard (single-signer) Schnorr signature scheme. F works as follows. F runs $\mathcal{Z}$ and $\mathcal{A}$, and it simulates the action of $\mathcal{S}$, i.e. the execution of $\pi_{\text{TSIG}}^{\text{SCHNORR}}$, as follows. It uses the SIP simulator for the key-generation phase of π_{DLKG} to force the output of the key-generation to be the given public-key y. Furthermore, to sign a message m, it asks the signing oracle for a signature (c, s) on m, it forces as above the outcome of S-1 to be $r = g^s/y^c$, and it uses a straightforward modification of the SIP simulator for the opening phase of π_{DLKG} to simulate the signing phase: the simulated $P_{j^\star}$ opens $K_{j^\star} C_{j^\star}^c$ to $s - \sum_{j \neq j^\star} s_j$ in step S-2 (rather than to $k_{j^\star} + c x_{j^\star}$), forcing the output of the signing phase to be the given signature (c, s). Additionally, whenever a message-signature pair (m, σ) is asked to be verified, F first checks whether m was never signed before and if σ is a valid signature on m. Once such a pair (m, σ) is found, F outputs that pair and halts. Similar to the proof of Theorem 3, one can show that if $\mathcal{A}$ does not corrupt the SIP then $\mathcal{Z}$ cannot distinguish between the real execution of $\pi_{\text{TSIG}}^{\text{SCHNORR}}$ (executed by the simulator $\mathcal{S}$) and the SIP simulation (executed by the forger F). Hence, by assumption on $\mathcal{Z}$ and $\mathcal{A}$, F outputs a signature on a message not signed by the signing oracle with non-negligible probability. □

We note that the above security argument concerns the weak case where the adversary outputs a signature on a *new* message. One can also show that the protocol is secure in stronger sense where the adversary attempts to yield one more signature on once signed messages.

5.4 A DL-Key Generation Variant with Threshold Shares

The protocol π_{DLKG} from Fig. 5 remains secure in the $(\mathcal{F}_{\text{HGEN}}, \mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g} \mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_h}, \mathcal{F}_{\text{SSMT}})$-hybrid model if the *threshold* shares $\xi_j = \sum_i x_{ij}$ of x are used in the opening phase rather than the additive shares x_j. Indeed, on receiving $(\textsf{private}, sid, x)$, the simulator $\mathcal{S}$ may simply adapt the secret-key shares ξ_i of the uncorrupted players such that the new ξ_i's form a sharing of x, hereby using the non-committing property of $\mathcal{F}_{\text{SSMT}}$. However, this attempt fails short when replacing $\mathcal{F}_{\text{SSMT}}$ in the execution of Pedersen's VSS by the (weakly secure) protocol π_{RNC} as we do, due to the DL condition in Lemma 1. Indeed, in order to adapt the ξ_i's, $\mathcal{S}$ needs to correspondingly adapt $P_{j^\star}$'s sharing of $x_{j^\star}$, such that it results in a sharing of $x - \sum_{j \neq j^\star} x_j$. However, $\mathcal{S}$ does not know the DL of (the encodings of) the adapted $x_{j^\star i}$'s (which are sent over $\mathcal{F}_{\text{SMT}}$) and thus π_{RNC} is committing. Recall that in the simulation of the original π_{DLKG}, $\mathcal{S}$ only needed to adapt $x_{j^\star}$, but not its sharing.

This problem, though, can be overcome by *randomizing* the secret-key shares ξ_i before publishing them in the opening phase. Concretely, every player P_j Pedersen VSSes 0, and then for every sharing of 0 every P_j adds the share he receives to ξ_j, resulting in a new share, $\tilde{\xi}_j$, of x. The opening phase is given in detail in Fig. 7. In the simulation, this randomization gives $\mathcal{S}$ the opportunity to make $\tilde{\xi}_1, \ldots, \tilde{\xi}_n$ a sharing of x without adapting the ξ_i's by letting $P_{j^\star}$ share $x - \sum_j x_j$ rather than 0. This can be done since the simulator knows the trapdoor to the commitment-key. Furthermore, using π_{RNC} to communicate the shares, this cannot be detected by the adversary $\mathcal{A}$ and thus by the environment $\mathcal{Z}$ (under the DDH assumption).

[Key-Generation Phase]

The key-generation phase is identical to that of π_{DLKG} from Fig. 5, except that each P_i stores $\xi_i = \sum_{j=1}^n x_{ji}$ as his secret-key share (rather than x_i), to which he is committed by $E_i = \prod_{j=1}^n E_{ji}$.

[Opening Phase]

O'-1. Each player P_j executes Pedersen VSS for secret 0 and default commitment 1 with regard to g and h: $\mathsf{PedVSS}^{j}_{g,h}(0) \to (o_{j1}, \ldots, o_{jn}, 0, 1, O_{j1}, \ldots, O_{jn})$. If a player P_j refuses then a default sharing of 0 is taken instead. Every P_j sets $o_j = \sum_{i=1}^n o_{ij}$. Write $O_j = \prod_{i=1}^n O_{ij}$.

O'-2. Each P_j opens publicly the commitment $E_j O_j$ to $\tilde{\xi}_j = \xi_j + o_j$. Secret-key x is reconstructed by Lagrange interpolation from the correctly opened $\tilde{\xi}_j$'s.

Fig. 7. Threshold DLKG protocol π'_{DLKG} with threshold shares.

Similar to Theorem 3 (whose full proof is given in Appendix D) but incorporating the above remarks, the following can be proven.

Theorem 5. *Implementing in the DLKG protocol π'_{DLKG} from Fig. 7 the functionalities $\mathcal{F}_{\text{HGEN}}$, $\mathcal{F}_{\text{SSMT}}$, $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ and $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_h}$ by π_{HGEN}, π_{RNC}, $\pi_{\text{XFVSS}}[g]$ and $\pi_{\text{XFVSS}}[h]$, respectively, results in a secure realization of $\mathcal{F}_{\text{DLKG}}$ against adaptive t-limited adversary for $t < n/2$ in the SIP UC model.*

From now on, when referring to protocol π'_{DLKG}, we mean π'_{DLKG} from Fig. 7 with the functionalities replaces by real-life protocols as specified in Theorem 5.

We also fix the following terminology. We say that the players "jointly generate a (degree-t) Pedersen sharing of 0" if they execute step O'-1 of protocol π'_{DLKG} (for a given commitment-key h). Similarly, we say that the players "jointly generate a (degree-t) Pedersen sharing of a random number" if they execute step O'-1 of protocol π'_{DLKG} modified in that every P_j shares a random $\rho_j \in \mathbb{Z}_q$ (rather than 0) so that the resulting sharing is a sharing of $\rho = \sum_j \rho_j$ (rather than of 0). We also consider the joint generation of degree-$2t$ Pedersen sharings, which can be achieved as above with obvious modifications.

5.5 Universally-Composable Threshold DSS-Signatures

In this section, we propose a threshold variant of the DSS signature scheme [20], provable secure in the UC framework.

The Original Scheme: In DSS, G_q is specified to be an order-q subgroup of $\mathbb{Z}_p^*$, such that a mapping $[\cdot] : G_q \to \mathbb{Z}_q$ can be defined by $[h] = h \bmod q$, where on the right hand side $h \in G_q \subset \mathbb{Z}_p^*$ is identified with its representation in $\{1, \ldots, p-1\}$. A DSS signature for message m under public-key $y = g^x$ consists of (r, s) such that $r = [g^{H(m)/s} y^{r/s}]$, where H is a cryptographic hash-function (SHA-1 according the DSS's specification). Such a signature is computed by the signer (who knows the secret-key x) by choosing $k \leftarrow \mathbb{Z}_q$ and computing

$$r = [g^k] \quad \text{and} \quad s = (H(m) + xr)/k\,.$$

The DSS signature scheme is believed to be secure (in the sense of existential unforgability against chosen message attacks). For our threshold DSS signature scheme it is important to note that it does not harm the security of the standard DSS signature scheme when $R = g^k$ is made available, as it can anyway be computed as $R = g^{H(m)/s} y^{r/s}$.

The UC Threshold Scheme: Our threshold version of the DSS signature scheme is illustrated in Fig. 8, and it uses the DLKG-variant π'_{DLKG} from Sect. 5.4. In its description we implicitly assume that corrupt players follow the signing phase (not necessarily honestly but) in such a way that they are not globally recognized as being corrupt. The easiest way to deal with a caught corrupt player is to restart the signing phase but without P_j taking part (and to have P_j excluded from any further invocations of the signing phase).

[Key-Generation Phase]

The players execute the key-generation phase of π'_{DLKG}, resulting in a public-key y, private *threshold* secret-key shares $\xi_1, \ldots, \xi_n$ with corresponding commitments $E_1, \ldots, E_n$, and commitment-key h.

[Signing Phase]

In order to sign a message m, the following steps are executed.

S-1. The players once more invoke the key-generation phase of π'_{DLKG}, but skipping the generation of h and taking h from the generation of y. Denote the output by $R = g^k$, and the corresponding (committed) threshold secret-key shares by $k_1, \ldots, k_n$. $r = [R]$ is the first part of the signature.

S-2. The players compute $s = (H(m) + xr)/k$, the second part of the signature, as follows.
- (a) The players jointly generate a degree-t Pedersen sharing $u_1, \ldots, u_n$ of a random element u, as well as two degree-$2t$ sharings $o_1, \ldots, o_n$ and $o'_1, \ldots, o'_n$ of 0.
- (b) Every player P_j broadcasts a commitment for the degree-$2t$ share $v_j = k_j u_j + o'_j$ of $v = ku$, proves its correctness (see comment in text), and publicly opens it (to v_j).
- (c) Every player P_j reconstructs v and puts $k_j^{\text{inv}} = u_j v^{-1}$ as his degree-t share of k^{-1}.
- (d) Every player P_j broadcasts the degree-$2t$ share $s_j = (H(m) + \xi_j r) k_j^{\text{inv}} + o_j$ of s and proves its correctness.
- (e) Every player P_j reconstructs s from $s_1, \ldots, s_n$.

Fig. 8. Threshold DSS-Signature Scheme $\pi_{\text{TSIG}}^{\text{DSS}}$

The proof for the correctness of v_j in step S-2 (b) is done by means of an off-the-shelf *witness-indistinguishable* honest-verifier (!) zero-knowledge proof of knowledge (based on the Pedersen commitments for k_j, u_j, o'_j and v_j), executed in parallel with every player P_i

acting as verifier, and the correctness proof is globally accepted if at least $t+1$ players P_i broadcast a confirmation that they accept P_j's (bilateral) proof. The corresponding holds for the correctness proof in step S-2 (d). The reason why we do not need to incorporate (fully) zero-knowledge proofs is that the SIP simulator does not need to simulate any proof as it knows a witness for every instance. The only security requirement needed from the proofs (besides soundness) is that it should remain hidden that the SIP simulator uses *different* witnesses (as in a real-life execution), which is covered by the witness-indistinguishability property.

Theorem 6. *Protocol $\pi^{\text{DSS}}_{\text{TSIG}}$ securely realizes $\mathcal{F}_{\text{TSIG}}$ against adaptive t-limited adversary for $t < n/2$ in the UC model, under the DDH assumption and under the assumption that the standard DSS signature scheme is secure.*

Proof. (Sketch) The simulator $\mathcal{S}$ simply executes *honestly* $\pi^{\text{DSS}}_{\text{TSIG}}$. In order to prove that this is a good simulation, we argue as in the proof of Theorem 4: The only way $\mathcal{Z}$ may see a difference is when $\mathcal{A}$ breaks the signature scheme, i.e., when a player provides at some point a valid signature on a message that has not been signed. However, if there exist $\mathcal{Z}$ and $\mathcal{A}$ that can enforce such an event with non-negligible probability, then there exists a forger F that breaks the existential unforgability against chosen message attacks of the standard (single-signer) DSS signature scheme. F works as follows. F runs $\mathcal{Z}$ and $\mathcal{A}$, and it simulates the action of $\mathcal{S}$, i.e. the execution of $\pi^{\text{DSS}}_{\text{TSIG}}$, as follows. It uses the SIP simulator for the key-generation phase of π'_{DLKG} to force the output of the key-generation to be the given public-key y. Furthermore, to sign a message m, it asks the signing oracle for a signatures (r, s) on m, it forces R in S-1 to be $R = g^{H(m)/s} y^{r/s}$ (such that $[R] = r$), and it honestly follows S-2, except that in step S-2 (a), in the joint generation of the Pedersen sharing $o_1, \ldots, o_n$ of 0, the SIP $P_{j^\star}$ deviates from the protocol and shares a value $\hat{o}$ (probably different than 0) chosen such that $s_1, \ldots, s_n$ in step S-2 results in a sharing of s: precisely, $P_{j^\star}$ shares $\hat{o} = s - (H(m) + \hat{x}r)/\hat{k}$ where $\hat{x}$ and $\hat{k}$ are the values to which the (simulated) ξ_j's and k_j's reconstruct, respectively. $P_{j^\star}$ can can do this as it knows the DL of h. Additionally, whenever a message-signature pair (m, σ) is asked to be verified, F first checks whether m was never signed before and if σ is a valid signature on m. Once such a pair (m, σ) is found, F outputs that pair and halts.

It remains to show that $\mathcal{Z}$ cannot distinguish the real execution of $\pi^{\text{DSS}}_{\text{TSIG}}$ (executed by $\mathcal{S}$) from the SIP simulation (executed by F), of course assuming that the SIP remains uncorrupted. Using techniques from the proof of Theorem 3 (specifically, by modifying $P_{j^\star}$'s executions of π_{RNC}), we may argue that $\mathcal{Z}$ has no information on the shares ξ_j, k_j, u_j, o_j and o'_j of the uncorrupted players P_j. As a consequence, $v_1, \ldots, v_n$ and $s_1, \ldots, s_n$ are random (degree-$2t$) sharings of a random value v respectively of $s = (H(m) + xr)/k$, in the real execution as well as in the SIP simulation. Furthermore, the corresponding correctness proofs are independent of the witness used and therefore identical in both cases. This completes the proof. □

5.6 Threshold Cramer-Shoup Cryptosystem

We would like to point out that the concurrent (erasure-free) threshold Cramer-Shoup scheme from [16], which *assumes* a correctly generated trapdoor commitment-key h, in combination with our protocol π_{HGEN} for generating h results in a threshold version of the Cramer-Shoup cryptosystem [10] which can be proven secure in the *SIP* UC model. Unfortunately, the techniques for achieving *fully* UC signatures (as in Sect. 5.3 and 5.5) do not translate to the threshold Cramer-Shoup (or any other) cryptosystem. The reason is the following. Because the simulator $\mathcal{S}$ has to produce an encryption of a default message in order to simulate the encryption phase (as it does not know the corresponding message), $\mathcal{S}$ has to "cheat" in the simulation of the decryption of such a ciphertext in order to enforce the outcome to be the correct message (rather than the default message). Such cheating however, seems to require a S! IP (or expensive fully non-committing encryptions). Note that such a problem does not occur for the signature schemes, where $\mathcal{S}$ can simply honestly follow the protocol, and the SIP is only needed to prove that this results in a "good" simulation.

6 Related Constructions

6.1 Adaptively-Secure Feldman VSS in Secure Channel Model

Here we present another adaptively secure Feldman VSS without erasure in $\mathcal{F}_{\text{SSMT}}$-hybrid model. This scheme is more efficient than the one in Fig. 1, but does not fit to the efficient realization of $\mathcal{F}_{\text{SSMT}}$ shown in Fig. 2 (essentially due to the DL condition of Lemma 1). Accordingly, the scheme is useful only in the secure channel model where $\mathcal{F}_{\text{SSMT}}$ is realized by a (possibly physical) secure channel.

Our construction is based on a special kind of commitment scheme, which we call a *trapdoor claw-free commitment scheme*. The claw-freeness means that for independently generated commitment keys K and K', it is infeasible to compute (s, r) and (s', r') such that $\mathsf{com}_K(s; r) = \mathsf{com}_{K'}(s'; r')$. And the trapdoor property means that besides the commitment key K the key-generation algorithm also outputs a trapdoor τ such that given τ, τ', s, r, s', one can efficiently compute r' such that $\mathsf{com}_K(s; r) = \mathsf{com}_{K'}(s'; r')$.

The well-known Pedersen commitment scheme $\mathsf{pcom}_{g,h}(s; r) = g^s h^r$ satisfies all of the above conditions, where the trapdoor τ is given by $\tau = \log_g(h)$: If there exists an algorithm $\mathcal{A}$ that outputs (s, r) and (s', r') such that $g^s h^r = g^{s'} h'^{r'}$ for randomly chosen h and h', one can use $\mathcal{A}$ to solve $\log_g Y$ for random Y by embedding it into h and h' as $h = Y$ and $h' = Y g^{\rho}$ with random ρ. On the other hand, given $\tau = \log_g(h)$ and $\tau' = \log_g(h')$, it is easy to compute r' for any s, r and s' such that $g^s h^r = g^{s'} h'^{r'}$, namely $r' = (s + \tau r - s')/\tau'$.

Our modification to (the original) Feldman's adaptively-secure VSS scheme simply replaces the encryption E in the set-up phase by a trapdoor claw-free commitment scheme com_K as above, whose commitment keys are given from each player. This is illustrated in Fig. 9.

X-1. Every P_j generates a commitment-key K_j and broadcasts it.

X-2. The dealer computes a commitment $A_j = \mathsf{com}_{K_j}(j; r_j)$ (with random r_j) for every $j \in \{1, \ldots, n\}$, and he chooses $\alpha_1, \ldots, \alpha_n$ as a random permutation over $1, \ldots, n$. Then, he broadcasts $A_1, \ldots, A_n$ ordered in such a way that A_j appears in α_j-th position, and he privately sends (α_j, r_j) to P_j. Simultaneously, the dealer starts to execute the sharing phase of the basic Feldman VSS, taking α_j as P_j's X-coordinate.

X-3. Each P_j identifies A_j in α_j-th position and accepts the assignment if $A_j = \mathsf{com}_{K_j}(j; r_j)$, and he accepts his share s_j if it satisfies the verification predicate (1) for the received α_j. Otherwise, he accuses the dealer who in turn has to publicly open A_j (to j) and announce s_j.

Fig. 9. Protocol π_{CFVSS} for adaptively secure Feldman-VSS in $\mathcal{F}_{\text{SSMT}}$-hybrid model.

An accusation by a player P_j is handled as in the basic Feldman VSS scheme, except that the dealer broadcasts (s_j, α_j, r_j) (rather than only s_j) and besides (1) it is also publicly verified whether A_j in α_j-th position satisfies $A_j = \mathsf{com}_{K_j}(j; r_j)$. Unlike to the original adaptively-secure Feldman VSS, where the set-up phase, i.e. the distribution of the α_j's, needs to be strictly separated from the actual sharing phase, these phases may interleave in this new variant. This saves one round of communication compared to the original scheme even if no accusation happens. The reconstruction phase remains unchanged. We denote this modified and parallelized Feldman VSS protocol by π_{CFVSS}. The security of π_{CFVSS} in the UC framework can be stated as follows.

Theorem 7. *Protocol π_{CFVSS} securely realizes $\mathcal{F}_{\mathrm{SVSS}}^{\mathsf{fcom}_g}$ in the $\mathcal{F}_{\mathrm{SSMT}}$-hybrid model (without erasures) against t-limited adaptive adversary for $t < n/2$.*

The proof is omitted since it can be done using essentially the same reasoning as for Proposition 1, except for the uniqueness of the assignments among honest players as claimed below.

Claim. For every honest P_i and P_j, $\alpha_i \neq \alpha_j$ holds with overwhelming probability.

Proof. A dealer who successfully assigns duplicate α_j to different honest players can be reduced to the following adversary $\mathcal{A}$: Adversary $\mathcal{A}$ is given $K_1, \dots, K_n$ generated by players. $\mathcal{A}$ is allowed to corrupt up to $t < n/2$ players and obtains corresponding trapdoor key τ_j on corruption. Then $\mathcal{A}$ specifies K_i and K_j and outputs (s, r), (s', r') that satisfy $\mathsf{com}_{K_i}(s; r) = \mathsf{com}_{K_j}(s'; r')$.

Now we show a reduction from $\mathcal{A}$ to breaking claw-free property as follows. Given K and K' for which we want to find a claw, guess $i^\star$ and $j^\star$ randomly from $\{1, \dots, n\}$ and set $K_{i^\star} = K$ and $K_{j^\star} = K'$. Generate (K_i, τ_i) for all other players. Then run $\mathcal{A}$ with $K_1, \dots, K_n$. Whenever $\mathcal{A}$ corrupts a player P_i, give τ_i. If $\mathcal{A}$ corrupts $P_{i^\star}$ or $P_{j^\star}$ then abort. If $\mathcal{A}$ happens to choose $K_{i^\star}$ and $K_{j^\star}$ and outputs correct (s, r), (s', r'), output (s, r), (s', r') as a claw. This reduction works only when the initial guess comes true. Accordingly, we have $\epsilon_{dup} < n^2 \epsilon_{claw}$ where ϵ_{dup} is the probability that duplicated α_j is successfully assigned and ϵ_{claw} the probability of finding a claw, respectively. Since ϵ_{claw} is assumed negligible, this proves the claim. □

Instantiating com by Pedersen's commitment scheme pcom, one can improve the reduction to be tight by exploiting the following facts. (1) K can be generated without knowing τ. Thus, it is not necessary to give τ to $\mathcal{A}$ on corruption. And (2) one can embed K and K' to all $K_1, \dots, K_n$ thanks to the random self-reducibility. By embedding K and K' respectively to the first and last $n/2$ keys, $\mathcal{A}$ eventually specifies K_i and K_j that embed K and K' with probability better than 1/2. This yields a claw. If either K or K' is embedded in both specified keys, it results in breaking the ordinary binding property of pcom, which is assumed to be successful only with negligible probability.

6.2 Pedersen's VSS as Committed VSS

Pedersen VSS, as given in Fig. 4, is a twin-base version of Feldman VSS. Instead of committing to secret s by g^s it uses so-called Pedersen commitment $\mathsf{pcom}_{g,h}(s; r) = g^s h^r$ with random r. As a result, the shared secret is statistically independent of the joint view of any t players if the shares are delivered in an unconditionally secure way. Accordingly, it seems that if the shares are delivered in a non-committing way, the scheme is in fact secure against adaptive adversaries: the simulator simply shares *some* secret, and then uses the trapdoor for the commitment scheme (the DL of h) in order to open the shares in an appropriate way in the reconstruction phase. This intuition, however, is correct only in a restricted sense. That is, if the output of the VSS functionality is limited to the reconstructed secret, while its commitment C (possibly chosen by a corrupt dealer) is not part of the output and thus cannot be used in external protocols. Hence, the original Pedersen VSS is secure as a non-committing VSS. On the other hand, if C is used in external protocols and thus needs to be part of the output of the VSS functionality, then the original Pedersen VSS is not secure against adaptive adversaries. The reason is that the corrupted dealer might choose C according to a strange and secret distribution that is unknown to the simulator, hence the simulator has to perform the simulation with given C to adjust to its distribution. Now the simulator encounters the same difficulty observed in the original Feldman's VSS.

Adaptively secure Pedersen VSS, π_{XPVSS}, as a committing VSS can be obtained by simply applying the randomized X-coordinate technique used for Feldman's VSS to the original Pedersen's VSS. As pointed out in Sect. 2.5, a technical subtlety is that the key h needs to be generated as part of the scheme (rather than considering it as given) in order to describe and prove secure the scheme correctly in the UC framework. (Even in the classical model, it is an important issue though often overlooked.) Optimally, h is uniformly distributed, which could for instance be achieved by the key-generation phase of $\mathcal{F}_{\text{DLKG}}$ (respectively π_{DLKG}). However, we claim that imposing uniform h is an overkill for the security of Pedersen's VSS. Indeed, Pedersen's VSS only requires that the dealer is not aware of $\log_g h$ while on the other hand the simulator is, which is exactly the property specified by $\mathcal{F}_{\text{HGEN}}$. Since $\mathcal{F}_{\text{HGEN}}$ is much less expensive than $\mathcal{F}_{\text{DLKG}}$ with our respective realizations, using $\mathcal{F}_{\text{HGEN}}$ is a more practical choice. Reflecting this argument, we introduce the combined functionality $\mathcal{F}_{\text{HGEN+SVSS}}^{\mathsf{pcom}_{g,?}}$ that captures Pedersen VSS in the UC framework. $\mathcal{F}_{\text{HGEN+SVSS}}^{\mathsf{pcom}_{g,?}}$ first generates key h simply by following the specification of $\mathcal{F}_{\text{HGEN}}$ and then executes the sharing and reconstruction phases of $\mathcal{F}_{\text{SVSS}}^{\mathsf{pcom}_{g,h}}$. We illustrate the scheme in Fig. 10 for completeness.

[Key Generation Phase]

The dealer invokes $\mathcal{F}_{\text{HGEN}}$ and every player obtains h.

[Sharing Phase]

P-1. The dealer initiates the protocol by sending *sharing* to every player.
P-2. Each P_j selects $\alpha_j \leftarrow \mathbb{Z}_q^*$ and sends it to the dealer by using $\mathcal{F}_{\text{SSMTwO}}$. The dealer replaces any α_j that happens to be 0 by $\alpha_j = 1$.
P-3. The dealer selects $f(X) = a_0 + a_1X + \cdots + a_tX^t \leftarrow \mathbb{Z}_q[X]$ and $f'(X) = b_0 + b_1X + \cdots + b_tX^t \leftarrow \mathbb{Z}_q[X]$ where $a_0 = s$. Let $r = b_0$. The dealer then computes and broadcasts $C = C_0 = g^s h^r$ and $C_k = g^{a_k} h^{b_k}$ for $k = 1, \ldots, t$, and he sends $s_j = f(\alpha_j)$ and $r_i = f'(\alpha_j)$ to P_i by $\mathcal{F}_{\text{SSMT}}$.
P-4. Each P_j verifies $g^{s_j} h^{r_j} = \prod_{k=0}^{t} C_k^{\alpha_j^k}$ and broadcasts *verified* if it holds. Otherwise, P_j initiates the accusation sub-protocol which is the same as that of Feldman VSS with obvious modification.

[Reconstruction Phase]

Each P_j broadcasts (α_j, s_j, r_j) and identifies $\mathcal{Q} \subseteq \{1, \ldots, n\}$ so that, for all $j \in \mathcal{Q}$, $g^{s_j} h^{r_j} = \prod_{k=0}^{t} C_k^{\alpha_j^k}$ holds. Then P_j reconstructs secret s by Lagrange interpolation with regard to $\mathcal{Q}$, and outputs s.

Fig. 10. Adaptively secure Pedersen VSS π_{XPVSS} in ($\mathcal{F}_{\text{SSMTwO}}$ $\mathcal{F}_{\text{HGEN}}$,$\mathcal{F}_{\text{SSMT}}$)-hybrid model.

Proposition 3. *Under the DL assumption, Protocol π_{XPVSS} in Fig. 10 securely realizes functionality $\mathcal{F}_{\mathrm{HGEN+SVSS}}^{\mathsf{pcom}_{g,?}}$ in the $(\mathcal{F}_{\mathrm{SSMTwO}}, \mathcal{F}_{\mathrm{HGEN}}, \mathcal{F}_{\mathrm{SSMT}})$-hybrid model against t-limited adaptive adversary for $t < n/2$.*

Using similar arguments as for Theorem 1, π_{XPVSS} gives rise to an efficient real-life version secure under the conditions (1) and (2) from Theorem 1.

The proof of Proposition 3 essentially follows the lines of the security proof for adaptive Feldman VSS (proof of Proposition 1) with obvious modifications. The main difference is that for the proof of Proposition 3 we need to argue that under the assumed hardness of computing DLs, the shares s_j provided by the corrupted players P_j during the reconstruction phase are consistent with the shares of the honest players (except with negligible probability). But this follows from the fact that if not, then the simulator can compute the DL of the generated key h and thus of any DL instance by embedding it into h.

6.3 Common-Random-String Generator

Note that if in π_{DLKG} the opening phase is invoked right after step K-2 (and steps K-3 and K-4 are skipped) then this modified protocol realizes a common-random-string (CRS) generator, which produces a random element from $\mathbb{Z}_q$. Name our modified protocol by π_{CRS} and the obvious CRS generator functionality by $\mathcal{F}_{\mathrm{CRS}}$. We have:

Theorem 8. *Protocol π_{CRS} securely realizes $\mathcal{F}_{\mathrm{CRS}}$ against adaptive t-limited adversary for $t < n/2$ in the SIP UC model.*

Similar protocols were used in [7, 16], though with the difference that the generation of h either required secure channels, or could not be proven secure without rewinding the adversary and thus disallows a UC-like security proof.

As shown in [7, 16], such a CRS generator allows the following application. Consider an execution of a standard three-move public-coin honest-verifier zero-knowledge proof protocol with challenge-space $\mathbb{Z}_q$ (like Schnorr's protocol for proving the knowledge of a DL or the Chaum-Pedersen protocol [9] for proving the equality of two DLs) where the first message and the answer are broadcast (by the prover) and the random challenge is generated jointly by an invocation of $\mathcal{F}_{\mathrm{CRS}}$. Such an execution still guarantees soundness, while it can be simulated without rewinding the possibly corrupt verifier. In [7, 16], this property holds under the assumption of a correctly generated h, while with our protocol π_{HGEN} it holds from scratch. The simulator can simply first produce an accepting transcript of the proof protocol using the honest-verifier zero-knowledge simulator, and then enforce the outcome of π_{CRS} to be the chosen challenge. This proof protocol can be useful in the construction of UC secure protocols; however, we stress that it is *not* a secure realization of the zero-knowledge functionality $\mathcal{F}_{\mathrm{ZK}}$ (modified to multiple verifiers). Secure realizations of $\mathcal{F}_{\mathrm{ZK}}$ in a distributed-verifier setting are sketched in Sect. 6.4.

6.4 Adaptively Secure Distributed-Verifier Proofs

In designing threshold cryptography, it is quite common to prove some relation (or knowledge) about committed witnesses in zero-knowledge manner. In the UC framework, however, zero-knowledge proofs are extremely expensive components: they are realized by combining a generic non-interactive zero-knowledge proof with a common-reference string generator, or UC-secure commitment scheme (which anyway needs common reference string) with generic zero-knowledge proof system for an NP-complete language such as Hamiltonian. They are generic and powerful, but cannot be directly used for practical subjects such as showing equality of discrete-logs or knowledge of a representation.

Combining our results with techniques developed in [1, 2], one can construct adaptively secure efficient distributed-verifier zero-knowledge proofs in universally composable way for many practical subjects. We illustrate a concrete example. Suppose that a prover needs to show that a triple, $(g^\alpha, g^\beta, g^\gamma)$ is in multiplicative relation $\alpha \cdot \beta = \gamma$, which is equivalent to showing that the triple is in DH. This can be done as follows. A prover shares α twice: once using the sharing phase of $\pi_{\text{XFVSS}}[g]$ and once using that of $\pi_{\text{XFVSS}}[g^\beta]$ with base g^β. Furthermore, in the second execution, the same sharing polynomial and X-coordinates as in the first execution are used. Hence the second execution is completed only by broadcasting a new commitment of the sharing polynomial, which is verified by the players by using the same share and X-coordinate received in the first execution. This guarantees that indeed the same secret, α, has been shared. Note that $(g^\beta)^\alpha$, supposed to be g^γ, is published in the second execution. Finally, the prover shares β (or γ) using the sharing phase of $\pi_{\text{XFVSS}}[g]$ with base g. If all sharing phases are accepted, the proof is accepted. Given $(g^\alpha, g^\beta, g^\gamma)$, $\mathcal{S}$ can simulate the prover by simulating the dealer in each execution of π_{XFVSS}. In the case of corrupt prover who completes the proof, $\mathcal{S}$ can extract α and β from the set of uncorrupted players. Hence the simulator can extract a witness (α, β) needed to invoke ideal zero-knowledge functionality.

The techniques of [1, 2] also apply to other commitment schemes that Feldman's, and allow to prove other relations as well like equality and additive and inverse relations among committed values. From these building blocks, one can even construct an adaptive distributed verifier proof for any NP relation by following the construction in [2].

Acknowledgments

We thank Jesper Buus Nielsen for suggesting to use spooling to overcome the problem addressed in Sect. 2.5. We also thank Stas Jarecki and the anonymous referees for their careful reading and invaluable comments.

References

1. M. Abe. Robust distributed multiplication without interaction. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 130–147. Springer-Verlag, 1999.
2. M. Abe, R. Cramer, and S. Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 206–223. Springer-Verlag, 2002.
3. M. Abe and S. Fehr. Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In M. Franklin, editor, *Advances in Cryptology — CRYPTO '04*, *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Annual Symposium on Foundations of Computer Science*, pages 136–145, 2001.
5. R. Canetti. On universally composable notions of security for signature, certification and authentication. In Cryptology ePrint Archive, Report 2003/239, 2003. `http://eprint.iacr.org`.
6. R. Canetti and M. Fischlin. Universally composable commitments (extended abstract). In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 2001.
7. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 98–115. Springer-Verlag, 1999.

8. R. Canetti and T. Rabin. Universal composition with joint state. In Cryptology ePrint Archive, Report 2003/047, 2002. `http://eprint.iacr.org`.
9. D. L. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.
10. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.
11. I. Damgård and J. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In M. Yung, editor, *Advances in Cryptology — CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer-Verlag, 2002.
12. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.
13. Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptively-secure distributed public-key systems. In J. Nesetril, editor, *European Symposium on Algorithms (ESA '99)*, volume 1643 of *Lecture Notes in Computer Science*, pages 4–27. Springer-Verlag, 1998.
14. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310. Springer-Verlag, 1999.
15. R. Gennaro, M. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *17th ACM Symposium on Principles of Distributed Computing*, 1998.
16. S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures (extended abstract). In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 221–242. Springer-Verlag, 2000.
17. J. Nielsen. Private communication, 2003.
18. T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer-Verlag, 1991.
19. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
20. U.S. Department of Commerce / National Institute of Standards and Technology. Digital signature standard. FIPS PUB 186, May 1994.
21. D. Wikström. A universally composable mix-net. In *Proceedings of the First Theory of Cryptography Conference (TCC'04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 315–335. Springer-Verlag, 2004.

A Proof of Proposition 1

First, we consider a statically corrupted dealer, i.e., P_d is corrupted by $\mathcal{A}$ before π_{XFVSS} starts. In this case, $\mathcal{S}$ corrupts $\tilde{P}_d$ to give its internal state to $\mathcal{A}$ and then follows all subsequent instructions from $\mathcal{A}$. If $\mathcal{A}$ initiates the sharing phase on behalf of corrupted dealer P_d, $\mathcal{S}$ simulates all the honest players as prescribed. Once the sharing phase is completed, each honest player P_j holds (α_j, s_j) that satisfies the verification predicate and each independently chosen α_j is different from others with overwhelming probability $> 1 - n^2/q$. $\mathcal{S}$ reconstructs secret s' (which may differ from P_d's actual input s) by interpolation, and then sends $(\mathsf{spool}, sid, s')$ and (share, sid) to $\mathcal{F}_{\mathrm{SVSS}}^{\mathsf{fcom}^g}$ under the name of $\tilde{P}_d$, and delivers the outputs to their intended players. Obviously, $\mathcal{S}$ succeeds in providing a perfect simulation except with negligible probability.

Next, we assume that the dealer remains uncorrupt forever. On receiving $(\mathsf{spooled}, sid, P_d, Y)$ from $\mathcal{F}_{\mathrm{SVSS}}^{\mathsf{fcom}^g}$, the simulator $\mathcal{S}$ works as follows. The item numbers correspond to those in Fig. 1.

F'-1. Simulate each honest player and $\mathcal{F}_{\mathrm{SSMT}}$ by following their prescription. If $\mathcal{A}$ corrupts a player, P_i, corrupt $\tilde{P}_i$ and give its internal state with α_i used in this step. Let $\{j_1, \ldots, j_k\}$ be set of identities of the players corrupted by the end of this step.

F'-2. Simulate dealer P_d as follows. Set $\tilde{\alpha}_i = \alpha_{j_i}$ for $1 \le i \le k$, and $\tilde{\alpha}_i \leftarrow \mathbb{Z}_q^*$ for $k+1 \le i \le t$. Then select $\tilde{s}_i \leftarrow \mathbb{Z}_q$ and compute $E_i = g^{\tilde{s}_i}$ for $i = 1, \ldots, t$. Let $C_0 = Y$ and broadcast $(C_0, C_1 \ldots, C_t)$ that satisfies the following system of equations.

$$\begin{pmatrix} E_1 \\ \vdots \\ E_t \end{pmatrix} = \begin{pmatrix} 1 \; \tilde{\alpha}_1^1 \ldots \tilde{\alpha}_1^t \\ \vdots \; \vdots \; \ddots \; \vdots \\ 1 \; \tilde{\alpha}_t^1 \ldots \tilde{\alpha}_t^t \end{pmatrix} \star \begin{pmatrix} Y \\ C_1 \\ \vdots \\ C_t \end{pmatrix} \tag{2}$$

Operator '$\star$' is a usual matrix multiplication but replacing addition/multiplication with multiplication/exponentiation. For instance, $E_1 = Y^1 {C_1}^{\tilde{\alpha}_1^1} \ldots {C_t}^{\tilde{\alpha}_1^t}$. From the inverse of the above Vandermonde matrix, $(C_1, \ldots, C_t)$ can be computed easily without solving discrete logarithms. Now, send $s_{j_i} = \tilde{s}_i$ to each corrupt P_{j_i}, and $s_j \leftarrow \mathbb{Z}_q$ to each honest P_j via $\mathcal{F}_{\text{SSMT}}$.
If $\mathcal{A}$ corrupts a player P_j at this stage, set $(\alpha_j, s_j) = (\tilde{\alpha}_{k+1}, \tilde{s}_{k+1})$. Then corrupt $\tilde{P}_j$ and give its internal state and (α_j, s_j) to $\mathcal{A}$. Then let $k := k + 1$.

F'-3. P_j is simulated to broadcast *verified* whatever it receives. For any accusation to P_d, simulate P_d and all honest players as prescribed. Note that honest players will never accuse P_d, and that P_d can answer all accusations correctly. If $\mathcal{A}$ corrupts a player, build the internal state of the player in the same way as in the previous step.

Observe that $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_t$ distributes as in the real execution (some follow the choice of corrupt players and others are chosen uniformly). Also, all $\tilde{s}_1, \ldots, \tilde{s}_t$ are chosen uniformly and other variables are in correct relation. Although α_j and/or s_j given to $\mathcal{A}$ as a part of corrupt P_j's internal state may be different from those actually sent to and received by P_j (namely when P_j is corrupted after the transmission of s_j), this is not observed by $\mathcal{A}$ due to the non-committing property of $\mathcal{F}_{\text{SSMTwO}}$ and $\mathcal{F}_{\text{SSMT}}$. Accordingly, all data observed by $\mathcal{A}$ and hence $\mathcal{Z}$ within π_{XFVSS} and $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$ with regard to the sharing phase distributes identically.

Considering the reconstruction phase, when $\mathcal{S}$ receives $(\textit{open}, sid, s)$ from $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$, it recovers the sharing polynomial, say $f(X)$, defined by points $(0, s)$, $(\tilde{\alpha}_1, \tilde{s}_1), \ldots, (\tilde{\alpha}_t, \tilde{s}_t)$. Then $\mathcal{S}$ computes $(\tilde{\alpha}_{t+1}, \tilde{s}_{t+1}), \ldots, (\tilde{\alpha}_n, \tilde{s}_n)$ where $\tilde{\alpha}_j \leftarrow \mathbb{Z}_q^*$ and $\tilde{s}_j = f(\tilde{\alpha}_j)$, and it assigns each of these newly computed points to each honest player so that they cast the points as their shares. If a player, P_j, is post-corrupted, $\mathcal{S}$ corrupts $\tilde{P}_j$ and hands its internal state together with the assigned point to $\mathcal{A}$.

As in the sharing phase, this simulated reconstruction phase provides $\mathcal{Z}$ a view that is perfectly indistinguishable from that in real-life model.

Finally, suppose that the dealer is corrupted during (of after) the execution of the protocol. To start with, on receiving $(\textit{spooled}, sid, P_d, Y)$ from $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$, the simulator $\mathcal{S}$ follows the above description (for the case of an uncorrupt dealer). If the dealer in this simulation is corrupted before step F-2, $\mathcal{S}$ corrupts the dealer in the ideal-process and obtains the secret s. Then $\mathcal{S}$ hands it to $\mathcal{A}$ and follows the instructions concerning corrupted P_d to the end of the protocol, while honestly simulating the uncorrupt players. When the sharing phase is completed, $\mathcal{S}$ recovers s' from the joint view of the honest majority and sends $(\textit{share}, sid, s')$ to $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$. If the dealer is corrupted after step F-2 but before the reconstruction takes place, $\mathcal{S}$ obtains s by corrupting the dealer in the ideal-process. Then, as above in the reconstruction phase, $\mathcal{S}$ reconstructs the sharing polynomial $f(X)$ from the $t + 1$ points, $(0, s), (\tilde{\alpha}_1, \tilde{s}_1), \ldots, (\tilde{\alpha}_t, \tilde{s}_t)$, computes the remaining points $(\tilde{\alpha}_{t+1}, \tilde{s}_{t+1}), \ldots, (\tilde{\alpha}_n, \tilde{s}_n)$ that are consistent with the polynomial, and hands everything over to $\mathcal{A}$ as part of the dealer's interal state. When the sharing phase is completed, $\mathcal{S}$ sends (share, sid, s) to $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$ (respectively $(\mathsf{share}, sid, 0)$ in case the sharing phase results in a default sharing of 0). In both cases, the reconstruction phase is simulated by letting every uncorrupt player P_j honestly follow the protocol, using his assigned point. Again, this simulation is perfect (and succeeds with overwhelming probability). □

B Proof of Lemma 1

Construction of $\mathcal{S}$: Hereafter, whenever element X is drawn from G_q, $\mathcal{S}$ first picks $x \leftarrow \mathbb{Z}_q$ and then computes $X = g^x$ so that all DLs are known. To remember this random choice with witness, we denote it by $X \overset{\mathsf{w}}{\leftarrow} G_q$.

We distinguish between the following cases. In case (1), the sender is (statically) corrupt, and in case (2), the sender remains uncorrupt.

Case (1): $\mathcal{S}$ corrupts $\tilde{P}_s$ statically, gives its internal state to $\mathcal{A}$, and follows subsequent instructions about P_s from $\mathcal{A}$. If corrupt P_s later initiates π_{RNC}, $\mathcal{S}$ simulates P_r honestly. If at some point during the execution of π_{RNC} $\mathcal{A}$ corrupts P_r, $\mathcal{S}$ corrupts $\tilde{P}_r$ and follows subsequent instruction about P_r from $\mathcal{A}$. On the other hand, if P_r remains uncorrupt and outputs a message m on completion of π_{RNC} (which might differ from the message $\mathcal{Z}$ sent to $\tilde{P}_s$), $\mathcal{S}$ sends $(\mathsf{send}, sid, \tilde{P}_r, m)$ to $\mathcal{F}_{\text{SMT}}$ (respectively $(\mathsf{spool}, sid, \tilde{P}_r, m)$ and (send, sid) to $\mathcal{F}_{\text{SSMT}}$) under the name of $\tilde{P}_s$ and delivers the output of $\mathcal{F}_{\text{SMT}}$ (respectively $\mathcal{F}_{\text{SSMT}}$) to its destination. This simulation is clearly perfect.

Case (2): On receiving $(sid, \tilde{P}_s, \tilde{P}_r)$ from $\mathcal{F}_{\text{SMT}}$, $\mathcal{S}$ starts π_{RNC}. Subsequent behavior of $\mathcal{S}$ depends on the timing that $\mathcal{A}$ corrupts receiver P_r as follows.

SA-0. In the initial step, $\mathcal{S}$ chooses $h \overset{\mathsf{w}}{\leftarrow} G_q$ for P_s and sends it to P_r.

SA-1. (Pre-corruption) If P_r is corrupted before y is sent off, $\mathcal{S}$ corrupts $\tilde{P}_r$ and gives $\mathcal{A}$ the internal state of $\tilde{P}_r$. On receiving $(\tilde{P}_s, m)$, $\mathcal{S}$ continues the simulation by simulating P_s honestly in the execution of π_{RNC} with message m.

SA-2. If pre-corruption does not happen, $\mathcal{S}$ honestly follows step A-1. Then it samples $u, v, c \overset{\mathsf{w}}{\leftarrow} G_q$ for P_s and sends (u, v, c) to P_r. (As a result, (h, u, v) distributes differently as in the real execution.)

SA-3. (Post-corruption) If P_r is corrupted after the transmission phase is completed, $\mathcal{S}$ gives $\mathcal{A}$ the internal state of $\tilde{P}_r$ as above. $\mathcal{S}$ also gives (z_1, z_2) that satisfies $y = g^{z_1} h^{z_2}$ and $c = Encode(m) u^{z_1} v^{z_2}$ where m is the message observed in the internal state of $\tilde{P}_s$. Such (z_1, z_2) can be computed easily (except with negligible probability $1/q$) since all the discrete-logs of h, y, u, v, c are known to $\mathcal{S}$ by construction, and that of $Encode(m)$ by assumption.

Reduction to DDH: We show that if there exists an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ such that $\text{REAL}_{\pi_{\text{RNC}},\mathcal{A},\mathcal{Z}} \not\approx \text{IDEAL}_{\mathcal{F}_{\text{SMT}},\mathcal{S},\mathcal{Z}}$ for the above simulator $\mathcal{S}$, then the DDH assumption does not hold. Precisely, we construct a distinguisher $\mathcal{D}$, which uses such $\mathcal{Z}$ and $\mathcal{A}$, and whose output distribution is given by $\text{REAL}_{\pi_{\text{RNC}},\mathcal{A},\mathcal{Z}}$ if $\mathcal{D}$'s input is chosen uniformly in DH and by $\text{IDEAL}_{\mathcal{F}_{\text{SMT}},\mathcal{S},\mathcal{Z}}$ if it is chosen uniformly in RND. Since the simulation differs from the real run only in step SA-2, we only consider case (2) in the following.

Let $(g^\alpha, g^\beta, g^\gamma)$ be a DDH instance chosen at random either from DH or from RND. On input $(g^\alpha, g^\beta, g^\gamma)$, $\mathcal{D}$ runs $\mathcal{Z}$ and $\mathcal{A}$ as well as (ideal-life) players $\tilde{P}_1, \ldots, \tilde{P}_n$ and the functionality $\mathcal{F}_{\text{SMT}}$, and it simulates an execution of π_{RNC} exactly as $\mathcal{S}$ does above, except for the following modifications. In step SA-0, $\mathcal{D}$ sets $h = g^\alpha$, and in step SA-2, $\mathcal{D}$ sets $u = g^\beta$,

$v = g^\gamma$ and $c = Encode(m)u^{z_1}v^{z_2}$, where m is the message $\tilde{P}_s$ received from $\mathcal{Z}$. Finally, $\mathcal{D}$ outputs the output bit of $\mathcal{Z}$.

First, consider the case $(g^\alpha, g^\beta, g^\gamma) \in \mathsf{DH}$, i.e., $\gamma = \alpha\beta$. Observe that for $r = \beta$, it holds that $u = g^r$, $v = h^r$ and $c = Encode(m)u^{z_1}v^{z_2} = Encode(m)y^r$. Thus, $\mathcal{D}$ simulates the players perfectly as in π_{RNC} and hence the output distribution of $\mathcal{Z}$ is identical to $\text{REAL}_{\pi_{\text{RNC}},\mathcal{A},\mathcal{Z}}$. Next, consider the case where $(g^\alpha, g^\beta, g^\gamma)$ is random in RND. Clearly, $\mathcal{D}$ produces the same simulation as $\mathcal{S}$, except if $(h, u, v) \in \mathsf{DH}$, which happens with probability $1/q$ and in which case $\mathcal{S}$ fails to handle a post-corruption. Accordingly, $\mathcal{D}$'s output distribution is statistically close to $\text{IDEAL}_{\mathcal{F}_{\text{SMT}},\mathcal{S},\mathcal{Z}}$. So, if $\mathcal{Z}$ distinguishes $\text{REAL}_{\pi_{\text{RNC}},\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{SMT}},\mathcal{S},\mathcal{Z}}$, then $\mathcal{D}$ distinguishes the respective uniform distributions over DH and RND with essentially the same advantage. $\square$

C Proof of Lemma 2

We focus on the claim concerning π_{SNCwO} and $\mathcal{F}_{\text{SSMTwO}}$. The proof of the security of π_{SNC} as a realization of $\mathcal{F}_{\text{SSMT}}$ can easily be extracted. For completeness, we illustrate protocol π_{SNCwO} in Fig. 11. Steps B-0 to B-3 are exactly the same as π_{RNC} except that all communication is done via broadcast (although we keep the terminology "send") and that the role of the sender and the receiver is reversed.

[Transmission Phase]

B-0. Receiver P_r chooses $h \leftarrow G_q$ and sends it to sender P_s.
B-1. P_s selects $z_1, z_2 \leftarrow \mathbb{Z}_q$, computes $y = g^{z_1}h^{z_2}$, and sends y to P_r.
B-2. P_r computes $u = g^r$, $v = h^r$ and $c = k\,y^r$, where $k \leftarrow G_q$, $r \leftarrow \mathbb{Z}_q$, and sends (u, v, c) to P_s
B-3. P_s recovers $k = c\,u^{-z_1}v^{-z_2}$ and sends $e = k\,Encode(m)$ to P_r.
B-4. P_r outputs $m = Decode(e\,k^{-1})$.

[Opening Phase]

C-1. P_s publishes z_1, z_2 and m.
C-2. Every player verifies whether $y = g^{z_1}h^{z_2}$, and, if true, computes $k = c\,u^{-z_1}v^{-z_2}$ and outputs $m = Decode(e\,k^{-1})$.

Fig. 11. Protocol π_{SNCwO} for sender non-committing transmission with opening.

Construction of $\mathcal{S}$: As in the proof of Lemma 1, whenever an element X is drawn from G_q, $\mathcal{S}$ first picks $x \leftarrow \mathbb{Z}_q$ and then computes $X = g^x$ so that all DLs are known. To remember this random choice with witness, we denote it by $X \overset{\mathsf{w}}{\leftarrow} G_q$.

We distinguish between the following cases:

(1) The receiver is statically corrupted (and the sender is statically or adaptively corrupted).
(2) The receiver is never corrupted and the sender is statically corrupted.
(3) The receiver is never corrupted and the sender is adaptively corrupted.

Case (1): $\mathcal{S}$ corrupts receiver $\tilde{P}_r$ at the beginning and gives its internal state to $\mathcal{A}$. On receiving $(\textsf{\textit{spooled}}, sid, \tilde{P}_s, \tilde{P}_r)$ from $\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{S}$ invokes π_{SNCwO} and simulates P_s honestly. If $\mathcal{A}$ corrupts P_s before e is sent off, $\mathcal{S}$ corrupts $\tilde{P}_s$ and gives $\mathcal{A}$ $\tilde{P}_s$'s internal state and all random choices used in the simulation of P_s so far. On the other hand, if P_s is not corrupted at step B-3, $\mathcal{S}$ gets back to the ideal-process and waits for message (*received*, $sid, \tilde{P}_s, m$) sent to the corrupt $\tilde{P}_r$. Once received, $\mathcal{S}$ continues the simulation by computing e correctly with this m. If P_s is corrupted at any further point, $\mathcal{S}$ corrupts $\tilde{P}_s$ and gives $\mathcal{A}$ $\tilde{P}_s$'s internal state and all random choices of P_s. If P_s does not get corrupted in the transmission phase and $\mathcal{S}$ receives (*sent*, $sid, \tilde{P}_s, \tilde{P}_r, m$) from $\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{S}$ publishes (z_1, z_2) chosen by P_s so that message m is recovered by following the protocol. Since there is no make-believe, $\mathcal{S}$ works perfectly like $\mathcal{A}$ from the viewpoint of $\mathcal{Z}$.

Case (2): $\mathcal{S}$ corrupts $\tilde{P}_s$, gives its internal state to $\mathcal{A}$, and follows subsequent instruction about P_s from $\mathcal{A}$. If corrupt P_s later initiates π_{SNCwO}, $\mathcal{S}$ simulates P_r honestly. If the transmission phase of π_{SNCwO} is completed and P_r outputs a message m (which might differ from the message $\mathcal{Z}$ sent to $\tilde{P}_s$, $\mathcal{S}$ sends (*spool*, $sid, \tilde{P}_r, m$) and (*send*, sid) to $\mathcal{F}_{\text{SSMTwO}}$ under the name of $\tilde{P}_s$ and delivers the output of $\mathcal{F}_{\text{SSMTwO}}$ to its destination. In the opening phase, if corrupt P_s publishes correct (z'_1, z'_2), $\mathcal{S}$ simulates all players to output corresponding $m' = Decode(e\,c^{-1}u^{z'_1}v^{z'_2})$ and sends (*open*, sid) to $\mathcal{F}_{\text{SSMTwO}}$ on behalf of corrupt P_s, upon which $\mathcal{F}_{\text{SSMTwO}}$ publishes m. Note that $y = g^{z'_1}h^{z'_2}$ and thus $Encode(m') = e\,c^{-1}u^{z'_1}v^{z'_2} = e\,c^{-1}(g^r)^{z'_1}(h^r)^{z'_2} = e\,c^{-1}y^r = Encode(m)$. Therefore, as in the above cases, $\mathcal{S}$ works perfectly.

Case (3): On receiving (*spool*, $sid, \tilde{P}_s, \tilde{P}_r$) from $\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{S}$ starts π_{SNCwO}. Subsequent behavior of $\mathcal{S}$ depends on the timing that $\mathcal{A}$ corrupts sender P_s as follows.

SB-0. In the initial step, choose $h \overset{\mathsf{w}}{\leftarrow} G_q$ for P_r.

SB-1. (Pre-corruption) If P_s is corrupted before (u, v, c) is sent off from P_r, $\mathcal{S}$ behaves as in Case (2) up to the end of the protocol execution.

SB-2. If pre-corruption does not happen, $\mathcal{S}$ continues simulation up to B-2 and select $(u, v, c) \overset{\mathsf{w}}{\leftarrow} {G_q}^3$ for P_r. (As a result, (h, u, v) distributes differently than in the real-life execution.)

SB-3. (Mid-corruption) If P_s is corrupted before e is sent off, $\mathcal{S}$ gives $\mathcal{A}$ the internal state of $\tilde{P}_s$. If P_r further receives $e \in G_q$ from corrupt P_s, $\mathcal{S}$ sets $m = Decode(e\,c^{-1}u^{z_1}v^{z_2})$ in step B-4, and sends (*send*, $sid, \tilde{P}_r, m$) to $\mathcal{F}_{\text{SSMTwO}}$ so that $\tilde{P}_r$ outputs m.

SB-4. If mid-corruption does not happen, $\mathcal{S}$ continues simulation by selecting $e \overset{\mathsf{w}}{\leftarrow} G_q$ for P_s in B-3. Once the transmission phase is done, $\mathcal{S}$ resumes the ideal-process and completes the transmission phase by handling the rest of messages from $\mathcal{F}_{\text{SSMTwO}}$ as specified.

SB-5. (Post-corruption) If P_s is corrupted after the transmission phase is completed, $\mathcal{S}$ corrupts $\tilde{P}_s$. It then computes (z_1, z_2) that satisfies $y = g^{z_1}h^{z_2}$ and $c = eEncode(m)^{-1}u^{z_1}v^{z_2}$ where m is the message observed in the internal state of $\tilde{P}_s$. Such (z_1, z_2) exists (unless $(h, u, v) \in \mathsf{DH}$, which happens with negligible probability $1/q$) and can be computed easily since all the discrete-logs of h, u, v, c, y, e are known to $\mathcal{S}$ by construction, and that of $Encode(m)$ by assumption. $\mathcal{S}$ gives such (z_1, z_2, m) to $\mathcal{A}$ with the internal state of $\tilde{P}_s$.

Finally, if corrupted P_s publishes correct (z'_1, z'_2) in the opening phase, $\mathcal{S}$ simulates all players to output $m' = Decode(e\,c^{-1}u^{z'_1}v^{z'_2})$ and sends (*open*, sid) to $\mathcal{F}_{\text{SSMTwO}}$. On the other hand, if P_s is not yet corrupted and (*sent*, sid, P_s, P_r, m) is sent from $\mathcal{F}_{\text{SSMTwO}}$, $\mathcal{S}$ computes (z_1, z_2) adjusted to this m in the same way as shown in SB-5 and publishes (z_1, z_2).

Concerning the question whether $m = m'$, we only need to look at the case of mid- or post-corruption, as in case of pre-corruption equality follows as in Case (2). We argue that (in case of mid- or post-corruption) $m \neq m'$ can happen only with negligible probability, under the DL assumption. Note that by construction $y = g^{z_1} h^{z_2}$ and $c = e\, Encode(m')^{-1} u^{z_1} v^{z_2}$, and by the correctness $y = g^{z'_1} h^{z'_2}$ and $c = e\, Encode(m')^{-1} u^{z'_1} v^{z'_2}$. Therefore, if $m \neq m'$ then $(z_1, z_2) \neq (z'_1, z'_2)$ and thus the two representations $g^{z_1} h^{z_2} = y = g^{z'_1} h^{z'_2}$ allow to compute the DL of h. It follows that the existence of $\mathcal{A}$ and $\mathcal{Z}$ that achieve $m \neq m'$ with non-negligible probability implies the existence of a algorithm that contradicts the DL assumption: simply adapt $\mathcal{S}$ in that it embeds a DL problem into h and, up to the point P_s gets corrupted, computes y, c and !e as $y = g^{z_1} h^{z_2}$, $c = k u^{z_1} v^{z_2}$ and $e = Encode(m)\, k$, where m is the message sent from $\mathcal{Z}$ to P_j. Note that this modification does not affect $\mathcal{Z}$'s view, except if $(h, u, v) \in \mathsf{DH}$ which happens with negligible probability $1/q$.

Reduction to DDH: Assuming that the unique opening is achieved as above, we now show that if there exists an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ such that $\text{REAL}_{\pi_{\text{SNCwO}},\mathcal{A},\mathcal{Z}} \not\approx \text{IDEAL}_{\mathcal{F}_{\text{SSMTwO}},\mathcal{S},\mathcal{Z}}$ for the above simulator $\mathcal{S}$, then the DDH assumption does not hold. Precisely, we construct a distinguisher $\mathcal{D}$, which uses such $\mathcal{Z}$ and $\mathcal{A}$, and whose output distribution is given by $\text{REAL}_{\pi_{\text{SNCwO}},\mathcal{A},\mathcal{Z}}$ if $\mathcal{D}$'s input is chosen uniformly in DH and by $\text{IDEAL}_{\mathcal{F}_{\text{SSMTwO}},\mathcal{S},\mathcal{Z}}$ if it is chosen uniformly in RND. Since the simulation differs from the real run only in step SB-2, we only consider Case (3) with Mid- or Post-corruption hereafter.

Let $(g^\alpha, g^\beta, g^\gamma)$ be an DDH instance chosen at random either from DH or from RND. On input $(g^\alpha, g^\beta, g^\gamma)$, $\mathcal{D}$ runs $\mathcal{Z}$ and $\mathcal{A}$ as well as (ideal-life) players $\tilde{P}_1, \ldots, \tilde{P}_n$ and the functionality $\mathcal{F}_{\text{SSMTwO}}$, and it simulates an execution of π_{SNCwO} exactly as $\mathcal{S}$ does above, except for the following modifications. That is, $\mathcal{D}$ sets $h = g^\alpha$ in SB-0, $u = g^\beta$, $v = g^\gamma$ and $c = k\, u^{z_1} v^{z_2}$ in SB-2, and $e = m\, k$ in SB-4. Here, k is chosen randomly as $k \overset{\mathsf{w}}{\leftarrow} G_q$ and m is the message $\tilde{P}_s$ received from $\mathcal{Z}$. Finally, $\mathcal{D}$ outputs the output bit of $\mathcal{Z}$.

Now, consider the case $(g^\alpha, g^\beta, g^\gamma) \in \mathsf{DH}$, i.e., $\gamma = \alpha\beta$. Observe that for $r = \beta$, it holds that $u = g^r$, $v = h^r$ and $c = k\, u^{z_1} v^{z_2} = k\, y^r$. Thus, $\mathcal{D}$ simulates the players perfectly as in π_{SNCwO} and hence the output distribution of $\mathcal{Z}$ is identical to $\text{REAL}_{\pi_{\text{SNCwO}},\mathcal{A},\mathcal{Z}}$. Next, consider the case $(g^\alpha, g^\beta, g^\gamma) \in \mathsf{RND}$, where γ is independent of α and β. In this case, (y, u, v, c, e) distributes uniformly over ${G_q}^5$ and $\mathcal{D}$ simulates the players in the same way as done by $\mathcal{S}$ (unless $(h, u, v) \in \mathsf{DH}$, which happens with negligible probability $1/q$) and the ensemble of the output distribution of $\mathcal{Z}$ is statistically close to $\text{IDEAL}_{\mathcal{F}_{\text{SSMTwO}},\mathcal{S},\mathcal{Z}}$. So, if $\mathcal{Z}$ distinguishes $\text{REAL}_{\pi_{\text{SNCwO}},\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{SSMTwO}},\mathcal{S},\mathcal{Z}}$, then $\mathcal{D}$ distinguishes the respective uniform distributions over DH and RND with essentially the same advantage. □

D Proof of Theorem 3

We start by proving π_{DLKG} from Fig. 5 secure in the $(\mathcal{F}_{\text{HGEN}}, \mathcal{F}_{\text{SSMT}}, \mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}, \mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_h})$-hybrid model.

On receiving (key, sid, y) from $\mathcal{F}_{\text{DLKG}}$, $\mathcal{S}$ simulates the generation phase by honestly following the protocol except that it simulates $\mathcal{F}_{\text{HGEN}}$ in step K-1 such that it knows the DL of h as well as for the following modification. After step K-2, $\mathcal{S}$ reconstructs x_j and r_j of any corrupt player P_j from the corresponding shares of the uncorrupt players and sets $C'_{j^\star} = y / \prod_{j \neq j^\star} g^{x_j}$ and $C''_{j^\star} = C_{j^\star} / C'_{j^\star}$. We show later the uniqueness of x_j and r_j. Then, instead of $P_{j^\star}$ sending $x_{j^\star}$ and $r_{j^\star}$ to the ideal functionalities $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ and $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_h}$, $\mathcal{S}$ simply makes them output $(\mathit{shared}, sid_{j^\star}, P_{j^\star}, C'_{j^\star})$ and $(\mathit{shared}, sid'_{j^\star}, P_{j^\star}, C''_{j^\star})$, respectively, as if they have received corresponding inputs from $P_{j^\star}$.

On receiving $(\mathit{private}, sid, x)$ from $\mathcal{F}_{\text{DLKG}}$, $\mathcal{S}$ simulates the opening phase honestly as prescribed, except that it makes $P_{j^\star}$ open $C_{j^\star}$ to $x - \sum_{j^\star \neq j} x_j$. This can be done as $\mathcal{S}$ knows the trapdoor to the commitment scheme.

We show the following properties of the simulation, of which each holds except with negligible probability under the DL assumption: (1) the uncorrupt players hold correct sharings of values x_j and r_j such that $g^{x_j} h^{r_j} = C_j$ for every player P_j, (2) the x_j's used to compute x in the opening phase coincide with the x_j's held by $\mathcal{S}$ (except for $P_{j^\star}$), and (3) the output of the simulated generation phase equals the y provided by $\mathcal{F}_{\mathrm{DLKG}}$.

Property (2) implies that the output of the simulated opening phase is indeed $x = \log_g y$; hence, the output behavior of the simulated protocol is as dictated by $\mathcal{F}_{\mathrm{DLKG}}$. Furthermore, it follows by inspection that the adversary $\mathcal{A}$'s view of the simulation is (statistically) indistinguishable from its view of a real protocol execution. Indeed, the difference only lies in $x_{j^\star}$'s (Pedersen) shares, of which $\mathcal{A}$ sees at most t. This then completes the security proof of $\mathcal{F}_{\mathrm{DLKG}}$ in the $(\mathcal{F}_{\mathrm{HGEN}}, \mathcal{F}_{\mathrm{SSMT}}, \mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}, \mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_h})$-hybrid-model.

Before showing the security of $\mathcal{F}_{\mathrm{DLKG}}$ in the real-life model when replacing the ideal functionalities by protocols as specified, we prove the claimed properties. (1) and (2) clearly hold with respect to any uncorrupt player P_j. With respect to corrupt P_j, (1) and (2) essentially follow from Lemma 3. If either of them does not hold, then one can use $\mathcal{A}$ (and $\mathcal{Z}$) to solve the discrete logarithm problem by embedding an instance h' of the discrete logarithm problem into h in the course of the simulation of $\mathcal{F}_{\mathrm{HGEN}}$ as shown in Section 5.1 and computing the DL of h (and thus of h') using Lemma 3. Concerning (3), if the output is not y then at least for one corrupt player P_j (for which x_j is not reconstructed in step K-4), the values x'_j and r'_j sent to $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ and $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_h}$, respectively, differ from x_j and r_j reconstructed by $\mathcal{S}$. However, as $g^{x_j} h^{r_j} = C_j = g^{x'_j} h^{r'_j}$, this allows to compute the DL of h and we can conclude as above.

Next, we show that we can safely replace $\mathcal{F}_{\mathrm{SSMT}}$ by π_{RNC} in the subprotocols $\mathsf{PedVSS}_{g,h}^{j}$ in π_{DLKG}. Consider the following simulator $\mathcal{S}'$ in the $(\mathcal{F}_{\mathrm{HGEN}}, \mathcal{F}_{\mathrm{VSS}}^{\mathsf{com}_g}, \mathcal{F}_{\mathrm{VSS}}^{\mathsf{com}_h})$-hybrid model. $\mathcal{S}'$ follows the specification of $\mathcal{S}$ above but executes protocol π_{RNC} whenever $\mathcal{S}$ calls for $\mathcal{F}_{\mathrm{SSMT}}$. Every instance of π_{RNC} is honestly executed except if the sender is $P_{j^\star}$ and the receiver, say P_i, is still uncorrupt when $P_{j^\star}$ comes to step A-2 of π_{RNC}. In that case, $P_{j^\star}$ computes u, v and c in step A-2 as $u, v \leftarrow G_q$ and $c = \mathit{Encode}(m) u^{z_1} v^{z_2}$, where m is the input to π_{RNC} and z_1 and z_2 are the values chosen by the simulated player P_i in step A-1. This way, uncorrupt P_i still receives the correct message m while the communication reveils no information on m. In order to show that this is a good simulation, we consider an artificial execution of π_{DLKG} with $\mathcal{F}_{\mathrm{SSMT}}$ replaced by π_{RNC}, with the same modification as above: if the sender of a message m is $P_{j^\star}$ and the receiver P_i is still uncorrupt when $P_{j^\star}$ comes to step A-2 of π_{RNC}, then $P_{j^\star}$ computes u, v and c in step A-2 as $u, v \leftarrow G_q$ and $c = \mathit{Encode}(m) u^{z_1} v^{z_2}$ (where we assume that P_i privately told $P_{j^\star}$ the values z_1, z_2 chosen in step A-1). By earlier observations that the difference between π_{DLKG} and its simulation only lies in the Pedersen shares of the uncorrupt players, it follows that $\mathcal{Z}$ cannot distinguish between the simulation provided by $\mathcal{S}'$ and the above artificial execution, as the communication of the shares of the uncorrupt players reveals no information. On the other hand, it is straightforward to show that if $\mathcal{Z}$ can distinguish the artificial execution from the real execution of π_{DLKG} with $\mathcal{F}_{\mathrm{SSMT}}$ replaced by π_{RNC}, then (together with $\mathcal{A}$) $\mathcal{Z}$ can be used to solve the DDH problem: one simply embeds a (randomized) DDH-problem instance $(g^\alpha, g^\beta, g^\gamma)$ into (h, u, v) of π_{RNC} and observes $\mathcal{Z}$'s output, similar as in the proof of Lemma 1 and 2.

Finally, using similar observations as for Theorem 2, it follows that π_{DLKG} securely realizes $\mathcal{F}_{\mathrm{DLKG}}$ in the real-life model when additionally replacing $\mathcal{F}_{\mathrm{HGEN}}$, $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_g}$ and $\mathcal{F}_{\mathrm{VSS}}^{\mathsf{fcom}_h}$ by π_{HGEN}, $\pi_{\mathrm{XFVSS}}[g]$ and $\pi_{\mathrm{XFVSS}}[h]$, respectively. This concludes the proof. $\square$