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] CWI, Amsterdam, The Netherlands
fehr@cwi.nl

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 UC model, which guarantees arbitrary composition as long as all protocols are executed by the same players. As an application, we propose a fully UC threshold Schnorr signature scheme.
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 sender non-committing encryption scheme, a distributed trapdoor-key generation protocol for Pedersen's commitment scheme, 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

* Research was carried out while at the Centre for Advanced Computing - Algorithms and Cryptography, Department of Computing, Macquarie University, Australia.

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 [14]. 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 [12]. Then Frankel *et al.* and Canetti *et al.* introduced in [11] 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 [13].

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. To the best of our knowledge, this is the first threshold scheme proven secure in the 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 Schnorr signature scheme.

Our DLKG protocol (and thus the threshold Schnorr signature scheme) 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 [10], 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 a sender non-committing encryption scheme, a new adaptively secure distributed trapdoor-key generation protocol for Pedersen's commitment scheme, as well as adaptively secure distributed-verifier zero-knowledge proofs, which all may very well find other applications. Finally, in the full version of this paper [3], we propose several additional applications and/or related adaptively-secure constructions of independent interest: a simple modification of Feldman's adaptively-secure VSS scheme which overcomes (1) and (2) above, though not (3), but is *fully* adaptively-secure, an adaptively secure version of Pedersen's VSS scheme as a *committed* VSS, a threshold version of the DSS signature scheme in the UC model, a threshold version of the Cramer-Shoup cryptosystem in the SIP UC model, and a common-reference-string generator with applications to zero-knowledge proofs in the UC model.

The paper is organized as follows. Section 2 reviews the model we are considering. It includes a short 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, universally-composable threshold cryptography and distributed-verifier proofs.

Due to space limitations, many definitions and proofs could only be sketched in this proceedings version of the paper; the formal treatment can be found in the full version [3].

2 Preliminaries

2.1 Communication and Adversary

We consider a *synchronized authenticated-link model* where a message from P_s to P_r is delivered within a constant delay and accepted by P_r if and only if it is sent from P_s to P_r. Moreover, we assume a broadcast channel with which every player sends a message authentically and all players receives the same message.

We consider a central adversary $\mathcal{A}$ which may corrupt players at will. Corrupting a player P_i allows $\mathcal{A}$ to read P_i's internal state and to act on P_i's behalf from that point on. In the non-erasure model, $\mathcal{A}$ additionally gains P_i's complete history. $\mathcal{A}$ is called *t-limited* if it corrupts at most t players. Furthermore, $\mathcal{A}$ is called *static* if it corrupts the players *before* the protocol starts, and $\mathcal{A}$ is called *adaptive* if it corrupts the players *during* the execution of the protocol, depending on what it has seen so far.

2.2 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]. We briefly sketch this framework here; for a more elaborate description we refer to the full version of the paper [3] or to the literature. The UC framework allows to define and prove secure cryptographic protocols as stand-alone protocols, while at the same time guaranteeing security in any application by means of a general composition theorem. In order to define a protocol π secure, it 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 privately and which honestly executes a number of specified commands. The UC security definition essentially requires that for every (real-life) adversary $\mathcal{A}$ attacking the protocol π, there exists an ideal-life adversary $\mathcal{S}$, also called *simulator*, which gets to attack the ideal-life scenario where only the players and $\mathcal{F}$ are present, such that $\mathcal{S}$ achieves "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 communicates in the real-life execution with $\mathcal{A}$ and in the ideal-life execution with $\mathcal{S}$. It is required that it cannot tell the difference between the real-life and the ideal-life executions, meaning that its respective outputs in the two cases are computationally indistinguishable.

As mentioned above, the UC framework provides a very general 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).

2.3 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 uncorrupt. 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 uncorrupt applies to (and either holds or does not hold) simultaneously for all protocols. With this limitation, the composition theorem essentially works as before. See also the full version of the paper [3].

2.4 Some Functionalities

We briefly introduce some functionalities we use throughout the paper. For formal descriptions and more elaborate discussions, we refer to [3].

Secure-Message-Transmission Functionalities: The secure-message-transmission functionality, as defined in [4], is denoted by $\mathcal{F}_{\text{SMT}}$. On receiving $(\textsf{send}, sid, P_r, m)$ from P_s, $\mathcal{F}_{\text{SMT}}$ sends (sid, P_s, m) to P_r and (sid, P_s, P_r) to the (ideal-life) adversary $\mathcal{S}$ and halts. If the length of m may vary, it is also given to $\mathcal{S}$.

Due to some subtle technicalities, $\mathcal{F}_{\text{SMT}}$ cannot be securely realized in a synchronized communication model against an active adversary. The reason is that in any (interactive) candidate realization π_{SMT}, the adversary $\mathcal{A}$ can corrupt the sender during the execution of the protocol and change the message m to be securely transmitted (or abort the protocol), while this cannot be achieved by $\mathcal{S}$. Indeed, once $\mathcal{F}_{\text{SMT}}$ is invoked it is always completed with the initial input (and the output is delivered to the receiver). To overcome this problem, we introduce *spooled* SMT. This is captured by $\mathcal{F}_{\text{SSMT}}$, which first spools the sent message m and only delivers the spooled message m (or a possibly different m in case of a corrupt P_s) when receiving another (the actual) send command from P_s. This allows $\mathcal{S}$ to change m after $\mathcal{F}_{\text{SSMT}}$ has been launched simply by corrupting P_s after the spool- but before the send-command.

$\mathcal{F}_{\text{SSMT}}$ can be realized over a public network using non-committing encryption as introduced by Canetti *et al.* in [6]. However, this is rather expensive as the best known schemes [9] still bear a ciphertext expansion $O(\kappa)$. Instead, our results are based on efficient though not fully adaptively secure realizations.

In our construction, we will also use an extended version of the $\mathcal{F}_{\text{SSMT}}$ functionality which allows P_s, in case of a dispute, to convince the other players of the message m sent to P_r. This is specified by $\mathcal{F}_{\text{SSMTwO}}$, spooled SMT *with opening*, which works as $\mathcal{F}_{\text{SSMT}}$ except that it additionally allows an open-command sent from P_s (and only from P_s), upon which it announces the transmitted message m to all players.

Committed-VSS Functionalities: 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). A commitment for a secret $s \in S_K$ is computed as $y = \mathsf{com}_K(s; r)$, 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 (with respect to commitment scheme com_K) is specified by functionality $\mathcal{F}_{\text{VSS}}^{\mathsf{com}_K}$, which sends $(\textsf{shared}, sid, P_d, \mathsf{com}_K(s; r))$ to all players and $\mathcal{S}$ on receiving ($\textsf{\textit{share}}, sid, (s,r)$) from P_d, the dealer, and later, on receiving ($\textsf{\textit{open}}, sid$) from $t+1$ distinct players, it sends ($\textsf{\textit{opened}}, sid, s$) to all players and $\mathcal{S}$.

Due to the same technical problem as above, if the dealer may be adaptively corrupted, we need to incorporate spooling into the committed VSS functionality: $\mathcal{F}_{\text{SVSS}}^{\mathsf{com}_K}$ first spools (s,r) (and gives $\mathsf{com}_K(s;r)$ to $\mathcal{S}$) before awaiting and executing the actual share-command (for the original or a new s).

We would like to mention that for certain candidate protocols π_{VSS} for committed VSS (with spooling), whose security relies 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 [3].

2.5 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 its 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.

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

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

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 $P_{j_1}, \ldots, P_{j_t}$. 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.[3] Suppose that honest P_j accuses the dealer, and that

[3] No dispute resolution procedure is shown in [10]. 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 $t < n/3$.

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 way to prove that the published α_i is different from the original assignment.

To efficiently settle such a dispute, digital signature scheme will be needed. That is, the dealer sends α_j 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 [3]. 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.

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. 2.4 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.

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).

[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 $f(X) = a_0 + a_1 X + \cdots + a_t X^t \leftarrow \mathbb{Z}_q[X]$ where $a_0 = s$ and computes $C_k = g^{a_k}$ for $k = 0, \ldots, t$ and broadcasts $(C_0, \ldots, C_t)$. For 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 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.

Proposition 1. *Protocol π_{XFVSS} shown in Fig. 1 securely realizes $\mathcal{F}_{\text{SVSS}}^{\mathsf{fcom}_g}$ 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 [3]. 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 $\mathbb{Z}_q \rightarrow G_q$ can be done by $m \mapsto M = m^2 \bmod p$, where $m \in \mathbb{Z}_q$ is identified with its representant 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 $\mathcal{F}_{\text{SMT}}$ (or $\mathcal{F}_{\text{SSMT}}$) 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, $\mathcal{F}_{\text{SMT}}$ can be realized (without spooling). We review the construction by [13] (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 the full version of the paper [3].

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 $c = Encode(m)\, y^r$, where $r \leftarrow \mathbb{Z}_q$, and sends (u, v, c) to P_r
A-3.	P_r computes $m = Decode(cu^{-z_1} v^{-z_2})$.

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

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

The assumption that the ideal-life adversary $\mathcal{S}$ is aware of the DL of $Encode(m)$ 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 $Encode(m)$. 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 $\mathcal{F}_{\text{SSMT}}$ 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 $Decode(k) \in \mathbb{Z}_q$), and then P_s sends $e = kEncode(m)$ to P_r, who computes m as $m = Decode(ek^{-1})$. 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 $m = Decode(ek^{-1})$.

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

The proof of Lemma 2 is similar to that of Lemma 1, although slightly more involved. For completeness, it is included in [3].

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. $\square$

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 DLKG protocol and a UC threshold Schnorr 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 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. If the trusted party is replaced with multi-party computation, as we usually do, the protocol should be designed not to use any common trapdoor commitment-key. In this section, we show a protocol that meets this requirement.

The players execute protocol π_{HGEN} 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 1 (Commitment-Key Generation Functionality: $\mathcal{F}_{\text{HGEN}}$).

1. *On receiving* $(\textsf{generate}, sid)$ *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* $(\textsf{com-key}, sid, h)$ *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.*

The proof is given in the full version of the paper. Essentially, on receiving (h', P_i) from $\mathcal{F}_{\text{HGEN}}$, $\mathcal{S}$ simulates the SIP $P_{j^\star}$'s call to $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ with input h', and it sends $\gamma = \sum_{j \neq j^\star} \chi_j$ to $\mathcal{F}_{\text{HGEN}}$.

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 $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ 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 $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ is never invoked in π_{HGEN}. Thus, the following holds.

Theorem 2. *Implementing $\mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g}$ in π_{HGEN} of Fig. 3 by π_{XFVSS} results in a secure realization of $\mathcal{F}_{\text{HGEN}}$ 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 [16]. 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 we then show a concrete example threshold scheme based on our DLKG protocol.

Definition 2 (Threshold DL Key Generation Functionality: $\mathcal{F}_{\text{DLKG}}$).
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 $\mathcal{F}_{\text{DLKG}}$ is illustrated in Fig. 5 below, and makes use of (ordinary) Pedersen's VSS scheme given in Fig. 4.

[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 $\mathcal{F}_{\text{SSMT}}$.

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: $\mathsf{PedVSS}_{g,h}(s) \rightarrow (s_1, \ldots, s_n, r, C)$

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 uncorrupt 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 $\mathsf{PedVSS}^j_{g,h}(s) \to (s_1, \ldots, s_n, r, C)$ 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 $s_1, \ldots, s_n, r, C$ generated as described in Fig. 4.

[Key-Generation Phase]

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

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

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

K-4. If P_i receives $(\textsf{shared}, sid_j, P_j, C'_j)$ and $(\textsf{shared}, sid'_j, P_j, C''_j)$, 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 $(\mathcal{F}_{\text{HGEN}}, \mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_g} \mathcal{F}_{\text{VSS}}^{\mathsf{fcom}_h}, \mathcal{F}_{\text{SSMT}})$-hybrid model.

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 $\mathcal{F}_{\text{SSMT}}$ in Pedersen's VSS is replaced by π_{RNC} as we do (due to the DL condition from Lemma 1). In [3] 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 $h^{r_j} = C''_j$ (see also Sect. 5.4).

Theorem 3. *Implementing in the DLKG protocol π_{DLKG} from Fig. 5 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.*

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 [13]: 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 the full version of the paper. 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 $\mathcal{F}_{\text{DLKG}}$. 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 $\mathcal{F}_{\text{DLKG}}$, $\mathcal{S}$ 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 uncorrupt 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 an example application of our DL-key generation protocol, we propose a threshold variant of Schnorr's signature scheme [15], provable secure in the UC framework. The scheme is illustrated in Fig. 6. Recall, 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 (in the single-signer variant), who knows the secret-key x, by choosing $k \leftarrow \mathbb{Z}_q$ and computing $r = g^k$, $c = H(m, r)$ and $s = k + cx$. Schnorr's signature scheme can be proven secure, in the sense of existential unforgability against chosen message attacks, in the *random oracle* model.

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.

Theorem 4. *Protocol π_{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 Schnorr signature scheme is secure.*

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

[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 π_{TSIG}

Proof. (Sketch) The simulator $\mathcal{S}$ simply executes *honestly* π_{TSIG}. 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. 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 π_{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 (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 $k_{j^\star} + cx_{j^\star}$), 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 π_{TSIG} (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. $\square$

5.4 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 DH, i.e. satisfies $\alpha \cdot \beta = \gamma$. 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 uncorrupt 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].

Acknowledgements

We thank Jesper Buus Nielsen for suggesting to use spooling to overcome the problem addressed in Sect. 2.4. 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 Cryptology ePrint Archive, Report 2004/119, 2004. `http://eprint.iacr.org`.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the* 42*nd 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, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Proceedings of the* 28*th annual ACM Symposium on the Theory of Computing*, pages 639–648, 1996.
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. I. Damgård and J. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450. Springer-Verlag, 2000.
10. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the* 28*th IEEE Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.
11. 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.
12. 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.
13. 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.
14. 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.
15. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
16. 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.