On Proactive Secret Sharing Schemes

Ventzislav Nikov[1] and Svetla Nikova[2,⋆]

[1] Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB, Eindhoven, the Netherlands
v.nikov@tue.nl
[2] Department Electrical Engineering, ESAT/COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Heverlee-Leuven, Belgium
svetla.nikova@esat.kuleuven.ac.be

Abstract. This paper investigates the security of Proactive Secret Sharing Schemes. We start with revision of the mobile adversary model of Herzberg's *et al.* imposing less restriction to the adversary. We first investigate the approach of using commitment to 0 in the renewal phase in order to renew the player's shares. In the considered model some well known computationally secure protocols (which use this approach) turns out to be vulnerable to a specific attack. We show that this type of attack is applicable also in the unconditional case. Then we extend the attack of D'Arco and Stinson to non-symmetric polynomials, which is applicable even in the mobile adversary model of Herzberg *et al.* Next the conditions for the security of a proactive scheme using this approach are shown. We also investigate another approach to add proactivity, namely using re-sharing instead of commitment to 0. Two protocols using this approach are described and it is shown that both are not secure against a mobile adversary. The main contribution of the paper is to show specific weaknesses, when a mobile adversary is considered.

1 Introduction

Verifiable secret sharing (VSS) schemes are secret sharing schemes (SSSs) dealing with possible misbehaving of the participants. Proactive security was first suggested by Ostrovsky and Yung in [14]. This concept was applied to the secret sharing schemes by Herzberg *et al.* in [9]. Basically the idea is that, if the information stored by the servers in order to share a given secret stays the same for all lifetime of the system, then an adversary can eventually break into a sufficient number of servers, to learn and destroy the secret. On the other hand,

⋆ The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT, IWT STWW project on Anonymity and Privacy in Electronic Services and Concerted Research Action GOA-MEFISTO-666 of the Flemish Government.

H. Handschuh and A. Hasan (Eds.): SAC 2004, LNCS 3357, pp. 308–325, 2005.

let the time is divided into periods. At the beginning of each period the information stored by the servers in a given time period changes, while the shared secret stays the same. Then the adversary probably does not have enough time to break into necessary number of servers. Moreover, the information he learns during the period t is useless during the period $t+i$, for $i = 1, 2, \ldots$. So, he has to start a new attack from scratch during each time period.

We revise the mobile adversary model from [9], imposing less restriction to the adversary. In the model of Herzberg's *et al.* the players corrupted during an update phase were considered corrupt for both (adjacent) periods. We propose a model in which these corrupt players are considered corrupt only in one of the adjacent periods. As a result in the new model some well known computationally secure protocols e.g. [9, 10, 8] became vulnerable to a specific attack, which we call attacks of first type.

The first unconditionally secure proactive VSS was proposed by Stinson and Wei [16]. In [12] a generalization of the scheme has been given, but D'Arco and Stinson [2, 3] showed that these two proactive schemes can be broken. We refer to their attack as of second type. The authors also proposed two new variations of the schemes to add proactive security to VSS, based on two different approaches, one using symmetric polynomials and another one using non-symmetric polynomials. However, in [12] an attack on the scheme with symmetric polynomials were described and slightly modified solution were proposed.

We next show that the first type attack is applicable also in the unconditional case in the considered model. Then we extend the second type of attack to the non-symmetric case. Note that the second type attack is successful even in the mobile adversary model of Herzberg's *et al.* We point out that a specific problem arises in the renewal phase, namely we need a distributed commitment protocol in which the committer is committed to 0 and the players are able to check that the commitment is indeed 0 without revealing their auxiliary shares. In order for this protocol to be secure against a mobile adversary we need to reduce the number of cheating players. The necessary and sufficient condition for security are consequently given in Theorem 3.

Last we investigate another approach [5, 6] to make an SSS proactively secure, namely using re-sharing protocol instead of commitment to 0 in order to renew and re-randomize the player's shares. We describe two protocols using this approach and show that both are subject to a modification of the second type attack. Our goal is to show specific weaknesses when mobile adversary is considered. Note that all unconditionally secure protocols we describe in this paper remain secure if the adversary is not mobile. Our aim throughout the paper is to learn more from the systems that fail in order to build systems that succeed.

2 Preliminary

2.1 Notations

Denote the *participants* (players) of the scheme by $P_i,\ 1 \leq i \leq n$, and the set of all *players* by $\mathcal{P} = \{P_1, \ldots, P_n\}$. Denote the *dealer* of the scheme by $\mathcal{D}$. The role

of the dealer is to share a secret s to all participants in the scheme. For the sake of simplicity we will consider only the threshold case in this paper. The simplest access structure Γ is called (k, n)-threshold if all subsets of players $\mathcal{P}$ with at least $k + 1$ participants are *qualified* to reconstruct the secret and any subset of up to k players are *forbidden* of doing it. Accordingly we will call a Secret Sharing Scheme (SSS) (k, n)-threshold if the access structure Γ associated with it is (k, n)-threshold.

2.2 Verifiable Secret Sharing Schemes

Verifiable Secret Sharing (VSS) schemes guarantee the robustness of the sharing and the detection of corrupt players. Informally, there are n players, some of them may be corrupt and deviate from the protocol. One of the players, the dealer, possesses a value s as a secret input. In the first stage, the dealer commits to a unique value $\widetilde{s}$ (no matter what corrupt players may do); moreover, $\widetilde{s} = s$ whenever the dealer is not corrupt. In the second stage, the already committed value $\widetilde{s}$ will be recovered by all good players (no matter what the corrupt players may do).

It is common to model cheating by considering an *adversary* $\mathcal{A}$ who may corrupt some of the players (up to k players). One can distinguish between *passive* and *active* corruption. Passive corruption means that the adversary obtains the complete information held by the corrupt players, but the players execute the protocol correctly. Active corruption means that the adversary takes full control of the corrupt players. Active corruption is strictly stronger than passive corruption. Both passive and active adversaries may be *static*, meaning that the set of corrupt players is chosen once and for all before the protocol starts, or *adaptive* meaning that the adversary can at any time during the protocol choose to corrupt a new player based on all the information he has at the time, as long as the total number of corrupt players is less or equal to k.

In the Appendix certain VSS schemes are given. Most of the used computationally secure schemes are based on Feldman's or Pedersen's VSS. We chose to consider only Feldman's scheme since it is simpler, but our attacks work against all these schemes. Also in the Appendix we present unconditionally secure VSS protocols, one based on symmetric and one based on asymmetric bivariate polynomials. We will refer to the unconditional secure sub-protocols described in the Detection phase also as "pair-wise" checking, for obvious reasons. These protocols ensure the consistency of the shares. We will refer to $h_v(0)$ and $g_v(0)$ as "true parts" of the shares since they are used to reconstruct the secret. The following result is classic for VSS theory.

Theorem 1. *A computationally secure (k, n)-threshold VSS exists if and only if $2k < n$. An unconditional secure (k, n)-threshold VSS exists if and only if $3k < n$.*

3 Computational Secure Proactive VSS Schemes

The concept of *proactive security* was introduced by Ostrovsky and Yung in [14] and applied by Herzberg *et al.* in [9] to secret sharing schemes. Proactive security refers to security and availability in the presence of a so-called *mobile adversary*. Herzberg *et al.* [9] have further specialized this notion to robust secret sharing schemes and have given a detailed efficient proactive secret sharing scheme.

Consider the following problem: if the information stored by the players in order to share a given secret stays the same for a long period of time (e.g. the lifetime of the system), then an adversary may gradually break into a sufficient number of players, to learn and destroy the secret. A way to address this problem is to divide the time into periods. At the beginning of each period the information stored by the players in that period changes, while the shared secret stays the same. The system is set up in such a way that the adversary does not have enough time to break into a required set of players. Moreover, the information that the adversary learns during a particular period is useless during later periods. So, he has to start a new attack from scratch during each time period.

Proactive security provides enhanced protection to long-lived secrets against a *mobile adversary*, i.e. the adversary which is allowed to potentially move among players over time with the limitation that it can only control some subset of players at a time unit. In fact, proactive security adds protection by "time diffusion". Namely, all shares are periodically refreshed. This renders useless the knowledge obtained by the mobile adversary in the past. Proactive systems also use robustness techniques to enhance availability by tolerating (detecting and correcting) malicious players. Moreover, it also allows *recoveries* of the previously corrupt players, by "removing" the adversary influence and restoring their (correct) information. This gives the system a *self-healing* nature. As a result, the system can tolerate a mobile adversary.

We will follow the settings of the schemes in [14, 9]. In general they coincide with the settings of the VSS except that we consider a more powerful adversary - a mobile one. In situations when the secret value needs to be maintained for a long period of time, in order to protect the secret against a mobile adversary, the life time is divided into time periods which are determined by the global clock. At the end of each time period the players engage in an interactive update protocol. The update protocol will not reveal the value of the secret. At the beginning of the next period the players hold new shares of the secret.

We assume that the adversary intruding player P_i is "removable", through a "reboot" procedure, when the adversary influence is detected. By "rebooting" the player we mean that the adversary's influence over this player is stopped and all player's information is erased. That is why after this procedure the correct share should be recovered. It is important to note that in proactive protocols some information (e.g. the check values, the old share, etc.) should be "erased". This operation, to be performed by honest players, is essential for the proactive security. Not doing so would provide an adversary that attacks a player at a given time period with information from a previous period that later could enable the adversary to break the system.

The following new phases *Recovery* and *Renewal* can be distinguished [9], compare to a VSS scheme. In [9] the update phase (also called update protocol) is separated from the time frames in a sense that if a player is corrupt during an update phase the authors consider it corrupt during both (adjacent) periods to that update phase. We consider the following

MOBILE ADVERSARY MODEL

At the beginning and at the end of the life time of the system we have *Share-Detection* respectively *Reconstruction*. At the end of each time period we have *Detection* followed by *Recovery* after that the next period begins with *Renewal*. Together Detection, Recovery and Renewal form an update phase, but we do not restrict additionally the adversary to corrupt players in this phase as in [9]. In fact the "rebooting" of the corrupt players finishes the current time frame and new time period begins.

The shares computed in period t for player P_u are denoted by using superscript (t), i.e. $s_u^{(t)}$, $h_u^{(t)}(x)$ or $g_u^{(t)}(y)$, $t = 0, 1, \ldots$. Dealer's polynomials corresponding to these shares are denoted by $f^{(t)}(x)$ and $f^{(t)}(x, y)$. Let us describe the Recovery and Renewal protocols given in [9].

We first briefly describe the idea how the player's shares are renewed at period $t = 1, 2, \ldots$. When the secret s is distributed as a value $f^{(t-1)}(0) = s$ of a k degree polynomial $f^{(t-1)}(x)$, we can update this polynomial by adding it to a k degree random polynomial $\delta^{(t-1)}(x)$, where $\delta^{(t-1)}(0) = 0$, so that $f^{(t)}(0) = f^{(t-1)}(0) + \delta^{(t-1)}(0) = s$. Thus we can renew the shares $f^{(t)}(\alpha_u) = f^{(t-1)}(\alpha_u) + \delta^{(t-1)}(\alpha_u)$ thanks to the linearity.

Renewal Phase:

1. Each player P_u plays the role of the dealer.
2. P_u runs the Share-Detection Phase of Feldman's VSS with a random polynomial $\delta_u(x) = \sum_{j=0}^{k} \delta_{u,j} x^j$ subject to $\delta_u(0) = 0$. The following broadcast values are used $A_{u,j} = \mathsf{g}^{\delta_{u,j}}$.
3. As a result of this Share-Detection Phase every player P_v has a temporary share $\delta_u(\alpha_v)$ if the player P_u is not blamed as a corrupt dealer.
4. Let A be the set of uncorrupt players.
5. Each player P_v updates its own share by performing
$$s_v^{(t)} = s_v^{(t-1)} + \sum_{u \in A} \delta_u(\alpha_v).$$
6. The new verification values are set $A_j^{(t)} = A_j^{(t-1)} \prod_{u \in A} A_{u,j}$.

Note that $\delta^{(t-1)}(x) = \sum_{u \in A} \delta_u(x)$ and that $A_j^{(t)}$ corresponds to the j-th coefficient in $f^{(t)}(x)$.

Now we describe the idea how the player's shares are recovered at period $t = 1, 2, \ldots$. Let the players in a set B are detected as corrupt and thus their shares should be recovered. Set $A = \mathcal{P} \setminus B$ to be the set of uncorrupt players. In general an analogous way to that used for re-randomization in the renewal phase is applied. First all corrupt players $P_v \in B$ are "rebooted". In order to

recover the share of player $P_v \in B$ every player $P_u \in A$ share a random k-degree polynomial $\delta_u(x)$, such that $\delta_u(\alpha_v) = 0$. By adding $\delta_u(x)$ to $f^{(t)}(x)$ (for $u \in A$) a new random polynomial $\delta(x)$ is obtained. Now the players $P_u \in A$ send their temporary shares $\delta(\alpha_u)$ to P_v, which allow him to recover the whole polynomial $\delta(x)$ and to compute his share $\delta(\alpha_v)$.

Theorem 2. *[9] A computationally secure (k, n)-threshold proactive VSS exists if and only if $2k < n$.*

3.1 The First Type of Attack

Proactive secret sharing [9] and proactive signature schemes [10] were introduced to cope with mobile adversary who may corrupt more than k servers during the life time of the secret. In both papers the proactive scheme is build on top of Feldman's VSS scheme [4]. In [8] the authors showed a specific attack against Feldman's VSS scheme and proposed a distributed key generation protocol build on top of Pedersen's VSS scheme [15]. The authors in [8] then claimed that their protocol is secure against a mobile adversary which can corrupt up to k players in given time frame.

In this section we will illustrate an attack against the renewal phase, in the schemes of Feldman, Pedersen and Genarro *et al.* We will show that even a passive, but mobile adversary can break these schemes in the considered model. For the sake of simplicity we will illustrate the attack only for Feldman's scheme.

Suppose that the attacker has corrupted a set B of players in some time frame $t-1$, i.e. he knows their shares $f^{(t-1)}(\alpha_u)$ for $u \in B$. All players $P_u \in B$ being detected as corrupt are "rebooted" and the new period t starts with the renewal phase, when all shares are updated. Now let the adversary corrupt k players (not in B) in this period and note that any k corrupt players can uniquely reconstruct the polynomial $\delta(x)$ since they have the additional information that $\delta(0) = 0$. Thus the adversary which gets information from k corrupt players in this period is able to compute the new player's shares $f^{(t)}(\alpha_u)$ for $u \in B$. Note that in this period the players $P_u \in B$ are no more corrupt. Therefore, incrementally breaking different sets of players the attacker is able to compute the secret. Actually the attacker needs to know only one share from the previous period which together with k player's shares from the current time frame will allow him to reconstruct the secret.

Note that the proposed attack applies if the renewal phase is considered as the beginning of the next period. However a slightly modified attack can be applied if we consider the renewal phase as the end of the previous period.

Therefore the first solution of Herzberg *et al.* allows even a passive, mobile adversary to break the scheme in the considered adversary model. Also most of the consecutive schemes, we will cite only some of them [9, 7, 10, 8], are subject to this kind of attack in the considered adversary model.

4 Unconditionally Secure Proactive VSS Schemes

The first unconditionally secure proactive VSS was proposed by Stinson and Wei [16]. A generalization of this scheme to general access structures has later been given in [11]. Recently D'Arco and Stinson [2, 3] showed that some existing unconditionally secure proactive schemes [11, 16] can be broken.

In [16, 2, 3] the authors consider different model in which all subsets of players with at least $k+1$ participants are qualified, but any subset of up to b ($b < k$) players is forbidden, where the restriction is due to the fact that some information is broadcast. So, we will consider (k, n) access structure where up to b ($b < k$) players are corrupt and will denote it by (b, k, n). Again we will present only Recovery and Renewal Phases. Recall that as a result of the previous phases all players maintain a set A of honest (not corrupt) players and possess shares $h_u^{(t)}(x)$. The shares $h_u^{(t)}(x)$ are derived from a symmetric polynomial $f^{(t)}(x, y)$ by setting $y = \alpha_u$. Set $B = \mathcal{P} \setminus A$ to be the set of corrupt players.

Recovery Phase:

1. Every corrupt player $P_v \in B$ is "rebooted".
2. Every good player P_u computes and sends to every corrupt player $P_v \in B$ a check-value $C_{u,v} = h_u^{(t)}(\alpha_v)$.
3. Upon receiving the data, P_v computes $h_v^{(t)}(x)$, such that $h_v^{(t)}(\alpha_u) = C_{u,v}$ holds for certain subset of honest, qualified players $P_u \in \widetilde{A}$, $\widetilde{A} \subseteq A$.
4. Player P_v sets $h_v^{(t)}(x)$ as his share.

Renewal Phase:

1. In this phase each player P_u plays the role of the dealer.
2. Each player P_u selects a random symmetric polynomial $\delta_u(x, y)$ of degree k, subject to $\delta_u(0, 0) = 0$.
3. Player P_u sends $\delta_{u;v}(x) = \delta_u(x, \alpha_v)$ to P_v for $1 \leq v \leq n$ and broadcasts $\delta_{u;0}(x) = \delta_u(x, 0)$.
4. Player P_v checks whether $\delta_{u;v}(0) = \delta_{u;0}(\alpha_v)$ and whether $\delta_{u;0}(0) = 0$.
5. If these relations are satisfied, then P_v computes and sends to P_w the usual check value $C_{u;v,w} = \delta_{u;v}(\alpha_w)$. Otherwise P_v broadcasts an accusation to P_u.
6. All players perform the (usual) pair-wise checking with accusations protocol. At the end they update the set of good players A.
7. Each player P_v updates his share by putting

$$h_v^{(t)}(x) = h_v^{(t-1)}(x) + \sum_{u \in A} \delta_{u;v}(x).$$

Set $\delta(x, y) = \sum_{u \in A} \delta_u(x, y)$, then $f^{(t)}(x, y) = f^{(t-1)}(x, y) + \delta(x, y)$ holds. Note that in Step 2 of the renewal phase additional information is broadcast, that we do not have in the standard Share-Detection phase. This information allows the players (in Step 4) to check that the value committed by P_u in Renewal phase is indeed 0. The latter ensures that the secret will not be changed.

4.1 The Second Type of Attack

Notice that the attack proposed in the previous section (the first type) is not applicable in this setting, since this attack is successfully mounted only when the number of corrupt players $b = k$. Obviously any k players using their temporary shares $\delta_v(x)$ together with the broadcast value $\delta_0(x)$ are able to compute $\delta(x, y)$. Therefore in case $b = k$ the first type of attack is applicable to the unconditional model.

But, it turns out that the broadcast information in the renewal phase allows the attacker to break the system even when $b < k$. We will demonstrate briefly the attack against the proactivity, proposed by D'Arco and Stinson [2], which we call second type attack.

Note that $\delta_{u;v}(0) = \delta_{u;0}(\alpha_v)$ holds. Suppose that the attacker has corrupted player P_v in some time frame $t-1$, i.e. he knows his share $h_v^{(t-1)}(x)$. Then P_v being detected as corrupt is "rebooted". In the renewal phase his share is updated by

$$h_v^{(t)}(x) = h_v^{(t-1)}(x) + \sum_{u \in A} \delta_{u;v}(x).$$

But since $\delta_{u;0}(x)$ is public the attacker is able to compute $\sum_{u \in A} \delta_{u;v}(0) = \sum_{u \in A} \delta_{u;0}(\alpha_v)$. Thus he knows the "true part" of the P_v's new share, namely $h_v^{(t)}(0) = h_v^{(t-1)}(0) + \sum_{u \in A} \delta_{u;v}(0)$. Recall that the knowledge of the "true part" of the shares is enough for reconstructing the secret. Therefore, incrementally breaking different sets of players the attacker is able to compute the secret.

4.2 Patching Stinson and Wei's Scheme

D'Arco and Stinson [2, 3] proposed two new variations of the unconditional schemes to add proactive security to VSS, based on two different approaches, one using symmetric polynomials and another one using asymmetric polynomials.

However, in [12] an attack on the proactive scheme with symmetric polynomials from [2] were described and a slightly modified scheme was proposed that solves this problem (see also [3]). For the sake of completeness we will provide here the solution for the symmetric case. The Recovery Phase is the same as in Stinson and Wei scheme.

Renewal Phase:

1. Each player P_u plays the role of the dealer.
2. Each player P_u selects a random symmetric polynomial $\delta_u(x, y)$ of degree $k-1$.
3. Player P_u sends $\delta_{u;v}(x) = \delta_u(x, \alpha_v)$ to P_v for $1 \le v \le n$.
4. Then P_v computes and sends to P_w the usual check value $C_{u;v,w} = \delta_{u;v}(\alpha_w)$.
5. All players perform the (usual) pair-wise checking with accusations protocol. At the end they update the set of good players A.
6. Each player P_v updates his share by putting

$$h_v^{(t)}(x) = h_v^{(t-1)}(x) + (x + \alpha_v) \sum_{u \in A} \delta_{u;v}(x).$$

Note that $\delta(x,y) = \sum_{u \in A} \delta_u(x,y)$ is a polynomial of degree $k-1$, but since at most b ($b < k$) players are corrupt the adversary can not compute $\delta(x,y)$.

4.3 D'Arco and Stinson's Scheme - The Asymmetric Case

In this section we will consider the second solution of D'Arco and Stinson (using asymmetric polynomials) proposed in [2, 3]. As in the previous section, we will consider (b,k,n) access structure where up to b ($b < k$) players are corrupt.

Recall that as a result of the previous phases all players maintain a set A of "good" (not corrupt) players and have shares $h_u^{(t)}(x)$ and $g_u^{(t)}(y)$. The shares are derived from an asymmetric polynomial $f^{(t)}(x,y)$ by setting $y = \alpha_u$ for $h_u^{(t)}(x)$ and by setting $x = \alpha_u$ for $g_u^{(t)}(y)$. Let $B = \mathcal{P} \setminus A$ be the set of corrupt players.

Recovery Phase:

1. Every corrupt player $P_v \in B$ is "rebooted".
2. Every good player $P_u \in A$ computes and sends to every corrupt player $P_v \in B$ the values $C_{u,v} = g_u^{(t)}(\alpha_v)$ and $D_{u,v} = h_u^{(t)}(\alpha_v)$.
3. Upon receiving the data, P_v computes $h_v^{(t)}(x)$ and $g_v^{(t)}(y)$, such that $h_v^{(t)}(\alpha_u) = C_{u,v}$, $g_v^{(t)}(\alpha_u) = D_{u,v}$ and $h_v^{(t)}(\alpha_v) = g_v^{(t)}(\alpha_v)$ hold for certain subset of honest, qualified players $P_u \in \widetilde{A}$ and $\widetilde{A} \subseteq A$.
4. Player P_v sets $h_v^{(t)}(x)$ and $g_v^{(t)}(y)$ as his shares.

Renewal Phase:

1. Each player P_u plays the role of the dealer.
2. Each player P_u selects a random polynomial $\delta_u(x,y)$ of degree k, subject to $\delta_u(0,0) = 0$.
3. Player P_u sends $h_{u;v}(x) = \delta_u(x, \alpha_v)$ and $g_{u;v}(y) = \delta_u(\alpha_v, y)$ to P_v for $1 \le v \le n$ and broadcasts $h_{u;0}(x) = \delta_u(x,0)$.
4. Player P_v checks whether $g_{u;v}(0) = h_{u;0}(\alpha_v)$ and $h_{u;0}(0) = 0$.
5. If the conditions are satisfied, then P_v computes and sends to P_w the (usual) check value $C_{u;v,w} = g_{u;v}(\alpha_w)$. Otherwise P_v broadcasts an accusation to P_u.
6. All players perform the usual pair-wise checking with accusations protocol and update the set of good players A.
7. Each player P_v updates his shares by putting

$$h_v^{(t)}(x) = h_v^{(t-1)}(x) + \sum_{u \in A} h_{u;v}(x)$$

$$g_i^{(t)}(y) = g_v^{(t-1)}(y) + \sum_{u \in A} g_{u;v}(y).$$

4.4 The Second Type of Attack - Asymmetric Case

We will show now that the above described protocol has a flaw. The idea is to apply a similar attack as described in [2, 3] for [16] (see also the previous section) but now applied to $g_u^{(t)}(y)$ instead of $h_u^{(t)}(x)$.

Suppose that the attacker has corrupted player P_v at some time frame $t-1$, i.e. he knows his shares $h_v^{(t-1)}(x)$ and $g_v^{(t-1)}(y)$. Then P_v being detected as corrupt is "rebooted". In the renewal phase his share is updated by

$$h_v^{(t)}(x) = h_v^{(t-1)}(x) + \sum_{u \in A} h_{u;v}(x) \qquad g_v^{(t)}(y) = g_v^{(t-1)}(y) + \sum_{u \in A} g_{u;v}(y).$$

Note that $g_{u;v}(0) = h_{u;0}(\alpha_v)$ holds. But since $h_{u;0}(x)$ is public the attacker is able to compute $\sum_{u \in A} g_{u;v}(0) = \sum_{u \in A} h_{u;0}(\alpha_v)$. Thus he knows the "true part" of the P_v's new share, namely $g_v^{(t)}(0) = g_v^{(t-1)}(0) + \sum_{u \in A} g_{u;v}(0)$. Note that the knowledge of the "true part" of the share of either $g_v(y)$ or $h_v(x)$ is enough for reconstructing the secret (see Remark 2 in the Appendix). Also it does not matter whether $h_{u;0}(x) = \delta_u(x, 0)$ or $g_{u;0}(y) = \delta_u(0, y)$ is broadcast since the attack is symmetric. Therefore incrementally breaking different set of players the attacker is able to compute the secret.

4.5 Conditions for Security of Proactive VSS

Now we are ready to refine the conditions for security of proactive VSS (Theorem 2), based on the considered approach to renew player's shares by sharing 0.

Theorem 3. *A computationally secure (b, k, n) (for $b < k$) proactive VSS exists if and only if $k + b < n$. An unconditionally secure (b, k, n) (for $b < k$) proactive VSS exists if and only if $k + 2b < n$.*

The first proactive protocols [9, 10] were applied to threshold access structures in the cryptographic setting. Since it was quite easy in that case to add the functionality of proactivity it was a common expectation that it would also be easy to add this functionality to all existing distributed protocols like VSS. But it turns out that a specific problem arises, namely in the renewal phase we need a distributed commitment protocol in which the committer is committed to 0 and the players are able to check that the commitment is indeed 0 without revealing their auxiliary shares. As a result of this specific problem several attacks against the Renewal phase that break the proactive security have been found. Thus the approach to refresh the shares by sharing 0 as a secret in the renewal phase seems to have a drawback, i.e. in order for the protocols to be secure against b cheating players we need to use polynomials of degree $k-1$ (instead of k) and hence we impose the requirement $b < k$.

Remark 1. The Renewal phase protocol in which 0 is shared as a secret is used as a stand alone sub-protocol in several other distributed protocols. Note that the weaknesses we pointed out here to these protocols arise only when mobile adversary is considered.

For the unconditional case all described attacks work even in the Herzberg *et al.* mobile adversary model.

5 Another Approach to Add Proactivity

Another approach to refresh (renew) the shares of the players is to re-share each share amongst the participants and then to combine the auxiliary shares in a special way. This approach was first applied to proactive SSS in [5, 6] divided there in two sub-protocols called sum-to-poly and poly-to-sum. These two sub-protocols together achieve the re-sharing goal. In general, every player first shares his own share (re-sharing) and then computes his new share as a certain linear combination of the auxiliary shares he receives from the other players, in such a way that at the end the players have new shares for the same secret as required in the renewal phase.

The approach of re-sharing the players shares is well known is SSS and it could be applied to change dynamically the access structure associated with the scheme. For example let $f(x)$ be k-degree polynomial such that $f(0) = s$ and let every player P_u has a share $s_u = f(\alpha_u)$. Then every player P_u chooses an ℓ-degree polynomial $g_u(x)$ such that $g_u(0) = s_u$, i.e. he re-shares his share sending auxiliary shares $g_u(\alpha_v)$ to player P_v. A set A of at least $k+1$ good players is determined. For such a set A there exist constants r_w (which depends only on A, but not on player's shares) such that $\sum_{w \in A} r_w s_w = s$. Now every player P_v combines the auxiliary shares he has received to compute his new share, i.e. $\widetilde{s_u} = \sum_{w \in A} r_w g_w(\alpha_v)$. It is easy to check that the new shares correspond to the same secret s and that the access structure is changed from (k, n) to (ℓ, n). Nearly the same protocol works in the computational secure VSS setting, e.g. Feldman's VSS.

On the other hand in the unconditionally secure VSS setting re-sharing and especially changing the access structure is more subtle. We will consider two protocols, which do not allow changing the access structure, since it is out of scope. Our goal is to show that the usual ways of doing re-sharing are not secure against a mobile adversary. First we will describe the straightforward way to re-share the shares. Then we will show that this protocol is not secure against a mobile adversary. Second we will describe another (more complex) protocol and will show that it is also not secure.

5.1 A Simple Re-sharing Protocol

Let us consider the protocol on Fig. 1 proposed in [13]. Every player P_u holds a share $h_u(x)$. The shares are derived from a symmetric polynomial $f(x, y)$ by setting $y = \alpha_u$. In the renewal phase the new shares are computed by $h_v^{(t)}(x) = \sum_{u \in A} r_u \, \delta_{u;v}(x)$. It is not difficult to verify that indeed **A.** We have new sharing for the same secret and **B.** The "symmetry" is not destroyed, i.e. the pair-wise check $h_v^{(t)}(\alpha_u) = h_u^{(t)}(\alpha_v)$ still holds for every u, v. The latter implies that there exists a symmetric polynomial $f^{(t)}(x, y)$ such that $f^{(t)}(0, 0) = s$ and $h_v^{(t)}(x) = f^{(t)}(x, \alpha_v)$.

Suppose now that the attacker has corrupted player P_v in some time frame $t-1$, i.e. he knows his share $h_v^{(t-1)}(x)$. Then P_v being detected as corrupt is "rebooted" and in the renewal phase his share is updated. Note that $\delta_{u;v}(0) =$

Re-sharing Phase:

1. Each player P_u re-shares the "true part" of his share, i.e. $h_u(0)$, by choosing a symmetric polynomial $\delta_u(x, y)$ of degree k such that $\delta_u(x, 0) = h_u(x)$.
2. Player P_u sends to P_v $(1 \leq v \leq n)$ temporary shares $\delta_{u;v}(x) = \delta_u(x, \alpha_v)$.
3. Each pair of players P_v and P_w exchange and then performs the usual pairwise-check: $\delta_{u;v}(\alpha_w) = \delta_{u;w}(\alpha_v)$.
4. In addition, each P_v checks his "true part" of the temporary share

$$\delta_{u;v}(0) = \delta_u(0, \alpha_v) = h_u(\alpha_v) = h_v(\alpha_u).$$

   The last equality is the pair-wise check in the VSS used to distribute the secret s. Note that this additional check ensures that player P_u really re-shares his share, i.e., he is an honest "dealer", and that player P_v has a consistent "true part" of the temporary share.
5. All players agree on a set of "good" players $A \in \Gamma$, which were not accused as corrupt dealers. Let r_u be the constants which correspond to players $P_u \in A$.
6. Each player P_v computes his new-share as follows:

$$h_v(x) \longleftarrow \sum_{u \in A} r_u \ \delta_{u;v}(x).$$

Fig. 1. A Simple Re-sharing Protocol [13]

$h_v(\alpha_u)$ holds. But since the attacker is able to compute $\sum_{u \in A} r_u \ \delta_{u;v}(0) = \sum_{u \in A} r_u \ h_v^{(t-1)}(\alpha_u)$, thus he knows the "true part" of the P_v's new share, namely $h_v^{(t)}(0) = \sum_{u \in A} r_u \ \delta_{u;v}(0)$. Recall that the knowledge of the "true part" of the shares is enough for reconstructing the secret. Therefore, again incrementally breaking different sets of players the attacker is able to compute the secret.

5.2 Re-sharing Protocol with Randomization

Another drawback of the protocol described in the previous section (on Fig. 1) is that the "true parts" of the shares are not re-randomized. That is why in this section we will avoid this drawback using a *commitment transfer protocol* [1] and proposing a kind of *commitment sharing protocol* [1] (see Fig. 2). As in the previous section we consider the following scenario. Every player P_u holds a share $h_u(x)$. The shares are derived from a symmetric polynomial $f(x, y)$ by setting $y = \alpha_u$. The protocol is on Fig. 2. Note that again the new shares are computed by $h_v^{(t)}(x) = \sum_{u \in A} r_u \ \delta_{u;v}(x)$. In the same way it is not difficult to verify that the conditions **A** and **B** are satisfied.

Suppose now that the attacker has corrupted player P_v in some time frame $t - 1$, i.e. he knows his share $h_v^{(t-1)}(x)$. Then P_v being detected as corrupt is "rebooted" and in the renewal phase his share is updated. Note that $\delta_{u;v}(0) = h_v^{(t-1)}(\alpha_u) - g_u(\alpha_v)$ and that $g_u(x)$ is public. Thus the attacker is able to compute $\sum_{u \in A} r_u \delta_{u;v}(0) = \sum_{u \in A} r_u \ (h_v^{(t-1)}(\alpha_u) - g_u(\alpha_v))$. He knows the "true part" of

the P_v's new share, namely $h_v^{(t)}(0) = \sum_{u \in A} r_u \ \delta_{u;v}(0)$. Therefore, again incrementally breaking different sets of players the attacker is able to compute the secret.

On the negative side we do not know secure perfect proactive VSS protocols, based on the considered approach to re-share the player's shares. On the positive side we can improve the conditions for security of proactive VSS (Theorem 3).

Re-sharing Phase:

1. Each player P_u re-shares the "true part" of his share, i.e. $h_u(0)$, by choosing a symmetric polynomial $\delta_u(x, y)$ of degree k.
2. Player P_u plays the role of the dealer executing Share-Detection phase.
3. As a result every player P_v posses a share $\delta_{u;v}(x)$ polynomial of degree k, if P_u is not blamed as a corrupt dealer.
4. In order to prove that the shared secret is indeed $h_u(0)$, P_u broadcasts a k-degree polynomial $g_u(x) = h_u(x) - \delta_u(x, 0)$. Note that if P_u is honest dealer then $g_u(0) = 0$ holds.
5. Each player P_v verifies that $g_u(0) = 0$ and that

$$g_u(\alpha_v) = h_u(\alpha_v) - \delta_u(\alpha_v, 0) = h_v(\alpha_u) - \delta_{u;v}(0).$$

   If these relations are satisfied he accepts his auxiliary share, otherwise an accusation against P_u is broadcast.
6. Let A be the set of uncorrupt players. Let r_u be the constants which correspond to players $P_u \in A$.
7. Each player P_v computes his new-share as follows:

$$h_v(x) \longleftarrow \sum_{u \in A} r_u \ \delta_{u;v}(x).$$

Fig. 2. Re-sharing Protocol with Randomization

Theorem 4. *A computationally secure (k, n) proactive VSS exists if and only if $2k < n$.*

6 A Remark on the General Access Structure Case

We first want to point out that all threshold protocols and attacks described in this paper can be easily generalized for the general access structure case using Monotone Span Programs (see [11, 12]). We choose not to do it just for the sake of simplicity, now we will only state the corresponding result to Theorem 3.

Denote the set of all subsets of $\mathcal{P}$ (i.e. the power set of $\mathcal{P}$) by $P(\mathcal{P})$. The set of qualified groups is denoted by Γ and the set of forbidden groups by Δ. The set Γ is called *monotone increasing* if for each set A in Γ also each set containing A is in Γ. Similarly, Δ is called *monotone decreasing*, if for each set B in Δ also each subset of B is in Δ. A monotone increasing set Γ can be efficiently described by

the set Γ^- consisting of the *minimal elements (sets)* in Γ, i.e. the elements in Γ for which no proper subset is also in Γ. Similarly, the set Δ^+ consists of the *maximal elements (sets)* in Δ, i.e. the elements in Δ for which no proper superset is also in Δ. The tuple (Γ, Δ) is called an *access structure* if $\Gamma \cap \Delta = \emptyset$. If the union of Γ and Δ is equal to $P(\mathcal{P})$ (so, Γ is equal to Δ^c, the complement of Δ), then we say that access structure (Γ, Δ) is *complete* and we denote it just by Γ. The adversary is characterized by a particular subset Δ_A of Δ, which is itself monotone decreasing structure. The set Δ_A is called *adversary structure* while the set Δ is called *privacy structure* The players which belong to Δ are also called *curious* and the players which belong to Δ_A are called *corrupt*. An (Δ, Δ_A)-adversary is an adversary who can (adaptively) corrupt some players passively and some players actively, as long as the set A of actively corrupt players and the set B of passively corrupt players satisfy both $A \in \Delta_A$ and $(A \cup B) \in \Delta$. For any two monotone *decreasing* sets Δ_1, Δ_2 operation *element-wise union* $\uplus$ is defined as follows: $\Delta_1 \uplus \Delta_2 = \{A = A_1 \cup A_2; A_1 \in \Delta_1, A_2 \in \Delta_2\}$.

Now we give a formal definition of a Monotone Span Program.

Definition 1. *A* Monotone Span Program *(MSP)* $\mathcal{M}$ *is a quadruple* $(\mathbb{F}, M, \boldsymbol{\varepsilon}, \psi)$, *where* $\mathbb{F}$ *is a finite field,* M *is a matrix (with* m *rows and* $d \leq m$ *columns) over* $\mathbb{F}$, $\psi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ *is a surjective function and* $\boldsymbol{\varepsilon} = (1, 0, \ldots, 0)^T \in \mathbb{F}^d$ *is called* target vector. *The size of* $\mathcal{M}$ *is the number* m *of rows and is denoted as size*$(\mathcal{M})$.

As ψ labels each row with a number i from $[1, \ldots, m]$ that corresponds to player $P_{\psi(i)}$, we can think of each player as being the "owner" of one or more rows. Let M_A denote the restriction of M to the rows i with $i \in A$. An MSP is said *to compute* a (complete) access structure Γ when $\boldsymbol{\varepsilon} \in \mathrm{im}(M_A^T)$ if and only if A is a member of Γ. We denote such an access structure by $\Gamma(\mathcal{M})$. We say that A is *accepted* by $\mathcal{M}$ if and only if $A \in \Gamma$, otherwise we say A is *rejected* by $\mathcal{M}$. In other words, the players in A can reconstruct the secret precisely if the rows they own contain in their linear span the target vector of $\mathcal{M}$, and otherwise they get no information about the secret.

Theorem 5. *Let* $\mathcal{M} = (\mathbb{F}, M, \boldsymbol{\varepsilon}, \psi)$ *be an MSP and* M *be an* $m \times d$ *matrix. Let* $\widetilde{\Delta}^c = \Gamma(\mathcal{M})$ *and let* $\widetilde{\Delta} \supseteq \Delta$. *Then there exist a perfect proactive VSS scheme secure against* (Δ, Δ_A)*-adversary if the following conditions are satisfied:*

1. $\mathrm{rank}(M_A) = d$, *for any group* $A \in \Gamma(\mathcal{M})^-$; *(Recovery)*
2. $\mathrm{rank}(M_B) \lneq d - 1$, *for any group* $B \in \Delta^+$; *(Renewal)*
3. $\mathcal{P} \notin \Delta_A \uplus \Delta_A \uplus \widetilde{\Delta}$. *(VSS)*

Note that the Vandermonde matrix is the matrix for MSP in the threshold case. Hence conditions 1. and 2. imply that $\widetilde{\Delta} \supsetneq \Delta$, i.e. $b < k$.

7 Conclusions

In this paper we have revised the mobile adversary model of Herzberg *et al.* and showed that the first scheme as well as most of the consecutive computationally

secure schemes are subject to a kind of attack in the new adversary model. We have shown that several unconditionally secure schemes can be broken when mobile adversary is considered (even in the Herzberg *et al.* adversary model), while the same protocols remain secure in case the adversary is not mobile. In conclusion we have shown several specific weaknesses. It is an open question whether we can do better than Theorem 3 (and Theorem 5), using for example the re-sharing approach instead of commitment to 0.

Acknowledgements

The authors would like to thank the anonymous referees for the valuable comments and remarks.

References

1. R. Cramer, I. Damgard, U. Maurer, General Secure Multi-Party Computation from any Linear Secret Sharing Scheme, *EUROCRYPT'2000*, LNCS 1807, Springer-Verlag 2000, pp. 316-334.
2. P. D'Arco, D. Stinson, On Unconditionally Secure Proactive Secret Sharing Scheme and Distributed Key Distribution Centers, *Manuscript*, May 2002.
3. P. D'Arco, D. Stinson, On Unconditionally Secure Robust Distributed Key Distribution Centers, *ASIACRYPT'2002*, LNCS 2501, Springer-Verlag 2002, pp. 346-363.
4. P. Feldman, A practical scheme for non-interactive verifiable secret sharing, *FOCS'1987*, pp. 427-437.
5. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, Proactive RSA, *CRYPTO'1997*, LNCS 1294, Springer-Verlag 1997, pp. 440-454.
6. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, Optimal-resilience proactive public-key cryptosystems, *FOCS'1997*, pp. 384-393.
7. S. Jarecki, Proactive Secret Sharing and Public Key Cryptosystems, *M.Sc. Thesis*, 1995, MIT.
8. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure Distributed Key Generation for Discrete-Log Based Cryptosystems, *EUROCRYPT'1999*, LNCS 1592, Springer-Verlag 1999, pp. 295-310.
9. A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, Proactive secret sharing or: How to cope with perpetual leakage, *CRYPTO'1995*, LNCS 963, Springer-Verlag 1995, pp. 339-352, (extended version 1998).
10. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, Proactive Public Key and Signature Systems, *ACM'1997 - Computer and Communication Security*, pp. 100-110.
11. V. Nikov, S. Nikova, B. Preneel, J. Vandewalle, Applying General Access Structure to Proactive Secret Sharing Schemes, *Proc. Benelux*, Springer-Verlag 2002, pp. 197-206, *Cryptology ePrint Archive:* Report 2002/141.
12. V. Nikov, S. Nikova, B. Preneel, J. Vandewalle, On Distributed Key Distribution Centers and Unconditionally Secure Proactive Verifiable Secret Sharing Schemes based on General Access Structure, *INDOCRYPT'2002*, LNCS 2551, Springer-Verlag 2002, pp. 422-437.

13. V. Nikov, S. Nikova, B. Preneel, Multi-Party Computation from any Linear Secret Sharing Scheme Unconditionally Secure against Adaptive Adversary: The Zero-Error Case, *ACNS'2003*, LNCS 2846, Springer-Verlag 2003, pp. 1-15.
14. R. Ostrovsky, M. Yung, How to withstand mobile virus attack, *PODC'1991*, pp. 51-59.
15. T. Pedersen, Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, *CRYPTO'1991*, LNCS 547, Springer-Verlag 1991, pp. 129-140.
16. D. Stinson, R. Wei, Unconditionally Secure Proactive Secret Sharing Scheme with combinatorial Structures, *SAC'1999*, LNCS 1758, Springer-Verlag 1999, pp. 200-214.

A Appendix

We first present Feldman's computational secure VSS protocol.

Sharing Phase:

Let s be a secret from a finite field $\mathbb{F} = \mathbb{Z}_{\mathtt{p}}$ and $\mathtt{g}$ is primitive element in $\mathbb{F}$. Each player P_u is associated publicly with different non-zero element $\alpha_u \in \mathbb{F}$.

1. Dealer $\mathcal{D}$ chooses a random polynomial $f(x)$ over $\mathbb{F}$ of degree k subject to the condition $f(0) = s$.
2. Each share s_u is computed by $\mathcal{D}$ as $s_u = f(\alpha_u)$ and then transmitted *secretly* to participant P_u.
3. Let $f(x) = \sum_{j=0}^{k} a_j x^j$. The dealer broadcasts the values $A_j = \mathtt{g}^{a_j}$ for $j = 0, 1, \ldots, k$.

Detection Phase:

1. Each player P_u verifies his own share by checking the following equation: $\mathtt{g}^{s_u} = \prod_{j=0}^{k} A_j^{\alpha_u^j}$. If the equation does not hold the player broadcasts an accusation to the dealer.
2. If there are more than k accusations to the dealer then $\mathcal{D}$ is blamed corrupt, and the protocol is stopped.

Reconstruction Phase:

1. Each player P_u broadcasts $f(\alpha_u)$.
2. Take $k+1$ broadcast values for which $\mathtt{g}^{f(\alpha_u)} = \prod_{j=0}^{k} A_j^{\alpha_u^j}$ holds.
3. Determine $\widetilde{f}(x)$ of degree at most k that passes through these points and output $\widetilde{f}(0)$.

Next we present two unconditional secure VSS protocols. The first one is based on symmetric bivariate polynomials.

Sharing Phase:

Let s be a secret from some finite field $\mathbb{F}$. Each player P_u is associated publicly with different non-zero element $\alpha_u \in \mathbb{F}$.

1. $\mathcal{D}$ chooses a random symmetric polynomial $f(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{i,j} x^i y^j$ over $\mathbb{F}$, where $a_{0,0} = s$ and $a_{i,j} = a_{j,i}$.
2. Then, for each player P_u, $\mathcal{D}$ sends $h_u(x) = f(x, \alpha_u)$ to P_u through a private channel.

Detection Phase:

1. Player P_u sends a check-value $C_{u,v} = h_u(\alpha_v)$ to P_v for $1 \leq v \leq n$, $(v \neq u)$.
2. Each player P_v checks whether $h_v(\alpha_u) = C_{u,v}$ for $1 \leq v \leq n$, $(v \neq u)$. If P_v finds that this is not true, then P_v broadcasts an accusation to P_u in the form $(v; u)$.
3. For each player P_w, who has been accused by a qualified group of players, the dealer must broadcast his share $h_w(x)$. Then each player again performs all relevant verifications on the values broadcast by the dealer and those known to him and accuses $\mathcal{D}$ if there is an inconsistency. The dealer defends himself by broadcasting back the share of the accusing player. This process continues until no new accusations are made.
4. Each player P_w computes the minimum subset $A \subseteq \mathcal{P}$, such that any ordered pair $(v; u) \in A \times A$ is not broadcast (i.e. is consistent). If $|A| \geq n - k$, then P_w accepts his share. Otherwise, P_w accuses the dealer.
5. If there are more than k accusations to the dealer then $\mathcal{D}$ is blamed corrupt, and the protocol is stopped.

Reconstruction Phase:

1. Each player $P_v \in A$ sends $h_v(x)$ to each $P_u \in A$.
2. After having received the polynomials $h_v(x)$, each $P_u \in A$ again applies non-interactive pair-wise checking for all received polynomials, namely: filling the consistency matrix with a 1 on position (v, w) if $h_v(\alpha_w) = h_w(\alpha_v)$ holds and with a 0 otherwise. Then P_u computes a subset of consistent shares $\widetilde{A} \subseteq A$, $\widetilde{A} \in \Gamma$.
3. Next, player P_u computes a polynomial $f_u(0, y)$, such that $f_u(0, \alpha_v) = h_v(0)$, for those v with $P_v \in \widetilde{A}$. Finally, the player P_u computes and outputs $s' = f_u(0, 0)$.

The second protocol is based on non-symmetric bivariate polynomials.

Sharing Phase:
Let s be a secret from some finite field $\mathbb{F}$. Each player P_i is associated publicly with different non-zero element $\alpha_i \in \mathbb{F}$.

1. $\mathcal{D}$ chooses a random polynomial $f(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{i,j} x^i y^j$, where $a_{i,j} \in \mathbb{F}$ and $a_{0,0} = s$.
2. Then, for each player P_u, $\mathcal{D}$ sends $h_u(x) = f(x, \alpha_u)$ and $g_u(y) = f(\alpha_u, y)$ to P_u through a private channel.

Detection Phase:
1. Player P_u checks whether $h_u(\alpha_u) = g_u(\alpha_u)$. If this condition is not satisfied he broadcasts an accusation on the dealer.
2. Next, player P_u sends a check value $C_{u,v} = g_u(\alpha_v)$ to P_v for $1 \leq v \leq n$, $(v \neq u)$.
3. Each player P_v checks whether $h_v(\alpha_u) = C_{u,v}$ for $1 \leq v \leq n$, $(v \neq u)$. If P_v finds that this is not true, then P_v broadcasts an accusation to P_u in the form $(v; u)$.
4. For each player P_w, who has been accused by a qualified group of players, the dealer must broadcast his share $h_w(x)$. Then each player again performs all relevant verifications on the values broadcast by the dealer and those known to him and accuses $\mathcal{D}$ if there is an inconsistency. The dealer defends himself by broadcasting back the share of the accusing player. This process continues until no new accusations are made.
5. Each player P_w computes the minimum subset $A \subseteq \mathcal{P}$, such that any ordered pair $(v; u) \in A \times A$ is not broadcast (i.e. is consistent). If $|A| \geq n - k$, then P_w accepts his share. Otherwise, P_w accuses the dealer.
6. If there are more than k accusations to the dealer then $\mathcal{D}$ is blamed corrupt, and the protocol is stopped.

Reconstruction Phase:
1. Each player $P_v \in A$ sends $h_v(x)$ and $g_v(y)$ to each $P_u \in A$.
2. After having received the polynomials $h_v(x)$ and $g_v(y)$, each $P_u \in A$ again applies non-interactive pair-wise checking for all received polynomials, namely: filling the consistency matrix with a 1 on position (v, w) if $h_v(\alpha_w) = g_w(\alpha_v)$ holds and with a 0 otherwise. Then P_u computes a subset of consistent shares $\widetilde{A} \subseteq A$, $\widetilde{A} \in \Gamma$.
3. Next, P_u computes a polynomial $f_u(0, y)$, such that $f_u(0, \alpha_v) = h_v(0)$, for those v with $P_v \in \widetilde{A}$. Finally, P_u computes and outputs $s' = f_u(0, 0)$.

Remark 2. Notice that the roles of the polynomials $h_v(x)$ and $g_v(y)$ are symmetric. Indeed, in the reconstruction phase a player P_u can also compute a polynomial $f_u(x, 0)$, such that $f_u(\alpha_v, 0) = g_v(0)$ for those v with $P_v \in \widetilde{A}$ and then he can again compute $s' = f_u(0, 0)$.