Serdica J. Computing **1** (2007), 337–364

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

ON PROACTIVE VERIFIABLE SECRET SHARING SCHEMES*

Ventzislav Nikov, Svetla Nikova, Bart Preneel

Abstract. This paper investigates the security of Proactive Secret Sharing Schemes. We first consider the approach of using commitment to 0 in the renewal phase in order to refresh the player's shares and we present two types of attacks in the information theoretic case. Then we prove the conditions for the security of such a proactive scheme. Proactivity can be added also using re-sharing instead of commitment to 0. We investigate this alternative approach too and describe two protocols. We also show that both techniques are not secure against a mobile adversary.

To summarize we generalize the existing threshold protocols to protocols for general access structure. Besides this, we propose attacks against the existing proactive verifiable secret sharing schemes, and give modifications of the schemes that resist these attacks.

ACM Computing Classification System (1998): D.4.6.
Key words: Secret Sharing Schemes, Proactive Security.

*The paper has been presented at the International Conference Pioneers of Bulgarian Mathematics, Dedicated to Nikola Obreshkoff and Lubomir Tschakaloff, Sofia, July, 2006.

The material in this paper was presented in part at the 11th Workshop on Selected Areas in Cryptography (SAC) 2004 [13].

1. Introduction. 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*, i. e., an adversary who is allowed to potentially move among players over time with the limitation that he can only control some subset of players (in $\Delta_{\mathcal{A}}$) at a given time unit. Herzberg *et al.* [9] have further specialized this notion to robust secret sharing schemes and have given a detailed efficient computationally secure 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, 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 latter periods. So, he has to start a new attack from scratch during each time period.

The goal of this paper is to show specific weaknesses when a mobile adversary is considered. The paper is organized as follows. In Section 2 we first introduce some notations and verifiable secret sharing schemes. Then we focus on the mobile adversary model. We first investigate the approach to renew a player's shares by sharing 0. Section 3 is devoted to computational secure schemes. Section 4 deals with unconditionally secure schemes and the certain kinds of attacks. A solution resisting known attacks is given in Section 5. The second approach to renew a player's shares, namely by re-sharing, is investigated in Section 6. Two solutions are proposed and shown to be insecure. Conclusions are presented in Section 7.

2. Preliminary.

2.1. Notations. Denote the *participants* 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 set of all subsets of $\mathcal{P}$ (i. e. the power set of $\mathcal{P}$) by $P(\mathcal{P})$ and 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. The set of *qualified* (to reconstruct the secret) groups is denoted by Γ and the set of *forbidden* (to reconstruct the secret) groups by Δ. The set Γ is *monotone increasing* since for each set A in Γ also each set containing A is in Γ. Similarly, Δ is *monotone decreasing*, since for each set B in Δ also each subset of B is in

Δ. Hence 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. For any two monotone *decreasing* sets Δ_1, Δ_2 operation *element-wise union* $\uplus$ is defined as follows: $\Delta_1 \uplus \Delta_2 = \{A = A_1 \cup A_2; A_1 \in \Delta_1, A_2 \in \Delta_2\}$.

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 to do it. Accordingly we will call a Secret Sharing Scheme (SSS) a (k, n)-threshold if the access structure Γ associated with it is (k, n)-threshold. Now we give a formal definition of a Monotone Span Program.

Definition 2.1 [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.

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* the 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. Hence when a set A is accepted by $\mathcal{M}$ there exists a so-called *recombination vector* (column) $\boldsymbol{\lambda}$ such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$. Notice that the vector $\boldsymbol{\varepsilon} \notin \mathrm{im}(M_B^T)$ if and only if there exists a vector $\boldsymbol{k} \in \mathbb{F}^d$ such that $M_B \boldsymbol{k} = \mathbf{0}$ and $\boldsymbol{k}_1 = 1$.

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. 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 *passively* and some *actively*. 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. 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 restricted. The adversary is characterized by a particular subset $\Delta_{\mathcal{A}}$ of Δ, which is itself monotone decreasing structure. The set $\Delta_{\mathcal{A}}$ 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 $\Delta_{\mathcal{A}}$ are called *corrupt*. An $(\Delta, \Delta_{\mathcal{A}})$-adversary ($(k, k_a)$-adversary in the threshold case) 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 $A \in \Delta_{\mathcal{A}}$ and $(A \cup B) \in \Delta$. The following result is classic for VSS theory:

Theorem 2.2. *A (k, n)-threshold VSS computationally secure against (k, k_a)-adversary exists if and only if $k + k_a < n$. An (k, n)-threshold VSS unconditional secure against (k, k_a)-adversary exists if and only if $2k + k_a < n$. A perfect VSS computationally secure against $(\Delta, \Delta_{\mathcal{A}})$-adversary exists if and only if $\mathcal{P} \notin \Delta_{\mathcal{A}} \uplus \Delta$. A perfect VSS unconditional secure against $(\Delta, \Delta_{\mathcal{A}})$-adversary exists if and only if $\mathcal{P} \notin \Delta_{\mathcal{A}} \uplus \Delta_{\mathcal{A}} \uplus \Delta$.*

2.3. The Settings. Proactive security provides enhanced protection to long-lived secrets against a *mobile adversary*. 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 consider a mobile $(\Delta, \Delta_{\mathcal{A}})$-adversary in the general case and (k, k_a)-adversary in the threshold case.

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 latter could enable the adversary to break the system.

We will follow the settings as presented in [14, 9] and for the sake of simplicity we will describe them only in the threshold case. The following two new phases *Recovery* and *Renewal* can be distinguished [9] in a Proactive scheme, compare to a VSS scheme. At the beginning and at the end of the life time of the system we have *Share-Detection* respectively *Reconstruction*. In general the settings coincide with those of the VSS except that we consider a more powerful adversary – a mobile one.

Mobile Adversary Model. 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 beginning of each time period the players engage in an interactive update protocol (also called *update phase*). The update protocol will not reveal the value of the secret. At the end of the update phase the players hold new shares of the secret.

The adversary can corrupt at most k (out of n) players at any moment during a time period (see Fig. 1A). But if a player is corrupted during an update phase, he is considered corrupted during both (adjacent to that update phase) periods (see Fig. 1B). Hence an extended time period begins and ends with update phase, i. e. any update phase belongs to two extended time periods and in each such time period the adversary can corrupt at most k players.

But actually there are more constrains on the time periods. Consider the case when the adversary corrupts k players in an update phase (e. g. $i+1$). Then the adversary can't corrupt more players neither in the time period $i+1$ not in time period i. Moreover in the update phases i and $i+2$ the adversary is also restricted to corrupt any additional player. Hence in the worst case the adversary can't corrupt more than k players in both periods i and $i+1$ (see Fig. 1C). Thus the strongest restriction to the adversary is that he can corrupt up to k players in any two consecutive time periods.

In [13] we proposed a modification to the mobile adversary model from [9], imposing less restriction to the adversary. Consider a model in which the corrupted during an update phase players are considered corrupted only in one of the adjacent periods. More specifically 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 (see Fig. 1D). The problem with this model is that in the renewal phase every player possesses both his old and new shares. This information from corrupted players in time period $i+1$ combined with the shares of other k corrupted players from time period i is enough to reconstruct the secret.

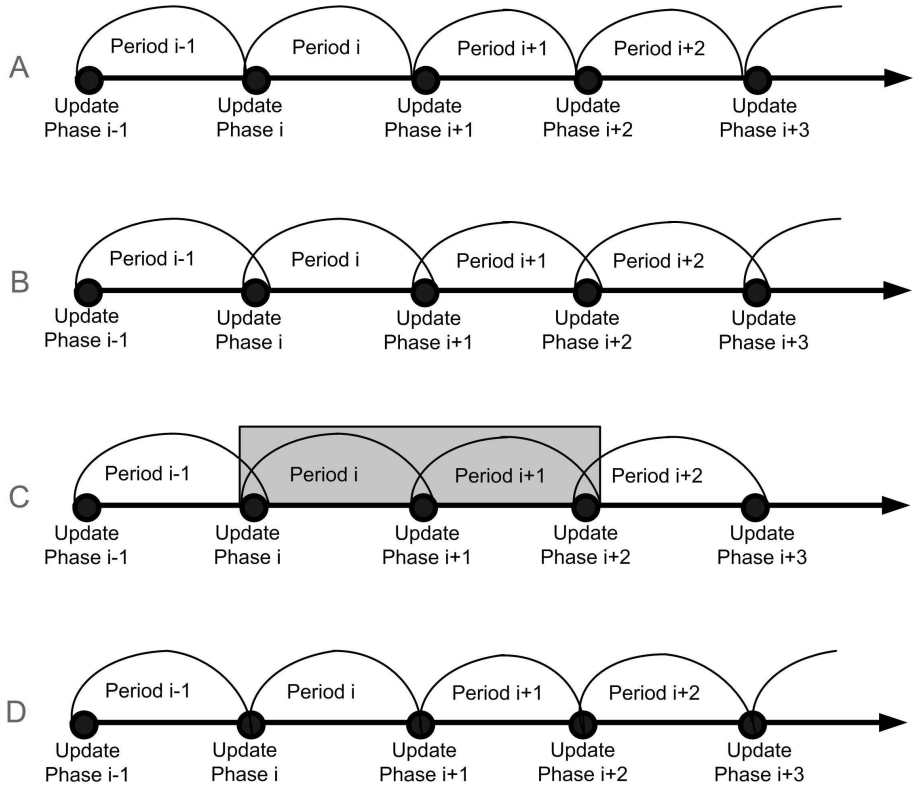

Fig. 1. Mobile Adversary Model

Road Map. In this paper we propose several types of attack to information theoretic case, call them second and third type of attacks. 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 5.1 and Theorem 5.2.

Last we investigate another approach [5, 6] to make an SSS proactively secure, namely using re-sharing instead of commitment to 0 protocol 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 similar 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.

The initialization (e. g. *Share* and *Detection* phases) and the *Reconstruction* phases are the same as in the VSSs described in the Appendix (see Figs 10, 11, 12,13). That is why further we will describe here only the *Renewal* and *Recovery* phases.

3. Computational Secure Schemes. Most of the used computationally secure schemes are based on Feldman's [4] or Pedersen's [15] VSS. We chose to consider only Feldman's scheme since it is simpler.

3.1. The Protocol. Herzberg *et al.* [9] propose a proactive scheme using Feldman and Pedersen VSS schemes (see Fig. 10). Let the shares computed in period t for player P_u be denoted by using superscript (t), i. e. $s_u^{(t)}$, $h_u^{(t)}(x)$ or $g_u^{(t)}(y)$, $t = 0, 1, \ldots$. Let the dealer's polynomials corresponding to these shares be 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 (see Fig. 2). Note that $\delta^{(t-1)}(x) = \sum_{u \in A} \delta_u(x)$ and that $C_j^{(t)}$ 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 $P_u \in A$ shares a random k-degree polynomial $\delta_u(x)$ such that $\delta_u(\alpha_v) = 0$. By adding $\delta_u(x)$ for $u \in A$ to $f^{(t)}(x)$ 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 3.1 [9]. *A computationally secure (k, n)-threshold proactive VSS exists if and only if $2k < n$.*

Proactive VSS [9]

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 $\delta_u^{(t-1)}(x) = \sum_{j=0}^{k} \delta_{u,j} x^j$ subject to $\delta_u^{(t-1)}(0) = 0$. The following broadcast values are used $C_{u,j} = \mathsf{g}^{\delta_{u,j}}$.
3. As a result of this Share-Detection Phase every player P_v has a temporary share $\delta_u^{(t-1)}(\alpha_v)$ 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
$$s_v^{(t)} = s_v^{(t-1)} + \sum_{u \in A} \delta_u^{(t-1)}(\alpha_v).$$
6. The new verification values are set $C_j^{(t)} = C_j^{(t-1)} \prod_{u \in A} C_{u,j}$.

Fig. 2. Proactive VSS [9]

4. Unconditionally Secure Schemes. We will refer to the unconditiopnal 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.

4.1. The First Protocols. The first unconditionally secure proactive VSS was proposed by Stinson and Wei [16]. Note that 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. Each player P_i is associated publicly with a non-zero element $\alpha_i \in \mathbb{F}$, so $|\mathcal{P}| < |\mathbb{F}|$. Recall that as a result of the previous phases all players maintain a set A of "good" (not corrupt) players and possess shares $h_i^{(t-1)}(x)$. The shares $h_i^{(t-1)}(x)$ are derived from a symmetric of degree k polynomial $f^{(t-1)}(x, y)$ by setting $y = \alpha_i$, (see Fig. 11 and Fig. 13). First we consider the threshold case and then we will generalize this protocol to general access structures case (see Fig. 3 and Fig. 4).

4.2. The Second Type of Attack. In Step 2 of the renewal phase additional information is broadcasted, that we do not have in the standard share-detection phase. This information allows the players to check that the value committed by P_e in the Renewal phase is indeed 0. Which ensures that the secret has not been changed. 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 $h_{e,i}^{(t-1)}(0) = h_{e,0}^{(t-1)}(\alpha_i)$ holds. D'arco and Stinson have proposed an attack against the proactivity [2] in the following way. Suppose that the attacker has corrupted player P_i in some time frame, i. e. he has share $h_i^{(t-1)}(x)$. Then P_i being detected as corrupt is "rebooted". In the renewal phase his share is updated by

$$h_i^{(t)}(x) \longleftarrow h_i^{(t-1)}(x) + \sum_{P_e \in A} h_{e;i}^{(t-1)}(x).$$

But since $h_{e;0}^{(t-1)}(x)$ is public information the attacker is able to compute the "true part" of the P_i's new share, namely

$$h_i^{(t)}(0) \longleftarrow h_i^{(t-1)}(0) + \sum_{P_e \in A} h_{e;i}^{(t-1)}(0) = h_i^{(t-1)}(0) + \sum_{P_e \in A} h_{e,0}^{(t-1)}(\alpha_i).$$

Recall that the knowledge of the "true part" of the share is enough for reconstructing the secret. Therefore, incrementally breaking different sets of players the attacker is able to compute the secret.

4.3. Patching the Scheme—Asymmetric Case. D'Arco and Stinson [2] proposed two variations of this scheme that both resist the attack described

Proactive VSS – Threshold Case [16]

Renewal Phase:

In this phase each player P_e plays the role of the dealer.

1. Each player $P_e \in A$ selects a random symmetric polynomial $\delta_e^{(t-1)}(x, y)$ of degree k, subject to $\delta_e^{(t-1)}(0, 0) = 0$.
2. Player P_e sends $h_{e;i}^{(t-1)}(x) = \delta_e^{(t)}(x, \alpha_i)$ to P_i for $1 \leq i \leq n$ and broadcasts $h_{e;0}^{(t)}(x) = \delta_e^{(t-1)}(x, 0)$.

Modified Detection:

1. Player P_i checks whether $h_{e;i}^{(t-1)}(0) = h_{e;0}^{(t-1)}(\alpha_i)$ and $h_{e;0}^{(t-1)}(0) = 0$.
2. If these relations are satisfied, then P_i computes and sends to P_j the usual check value $h_{e,i}^{(t-1)}(\alpha_j)$. Otherwise P_i broadcasts an accusation to P_e.
3. All players perform the pair-wise checking with accusations protocol. At the end they update the set of good players A.
4. All players update their shares by putting

$$h_i^{(t)}(x) \longleftarrow h_i^{(t-1)}(x) + \sum_{P_e \in A} h_{e;i}^{(t-1)}(x).$$

Recovery Phase:

1. Every corrupt player $P_j \notin A$ is "rebooted".
2. Every good player $P_i \in A$ computes and sends to every corrupt player P_j the check value $h_i^{(t-1)}(\alpha_j)$.
3. Upon receiving the data, P_j computes $h_j^{(t-1)}(x)$, such that $h_j^{(t-1)}(\alpha_i) = h_i^{(t-1)}(\alpha_j)$ holds for some subset of honest, qualified players $P_i \in B$, $B \subseteq A$.
4. Player P_j sets $h_j^{(t-1)}(x)$ as his share.

Fig. 3. Proactive VSS—Threshold Case [16]

Proactive VSS—General Case [11]

Renewal Phase:

In this phase each player P_e plays the role of the dealer.

1. Each player $P_e \in A$ selects a random symmetric $d \times d$ matrix $R^{(e)}$, subject to 0 in its upper left corner.
2. Player P_e sends $U_i^{(e)} = M_i R^{(e)}$ to P_i for $1 \leq i \leq n$ and broadcasts $U_0^{(e)} = \boldsymbol{\varepsilon}^T R^{(e)}$.

Modified Detection:

1. Player P_i checks whether $U_i^{(e)}\boldsymbol{\varepsilon} = M_i(U_0^{(e)})^T$ and $< U_0^{(e)}, \boldsymbol{\varepsilon}^T >= 0$.
2. If these relations are satisfied, then P_i computes and sends to P_j the usual check matrix $C_{ij}^{(e)} = U_i^{(e)} M_j^T$. Otherwise P_i broadcasts an accusation to P_e.
3. All players perform the pair-wise checking with accusations protocol. At the end they update the set of good players A.
4. All players update their shares by putting

$$U_i \longleftarrow U_i + \sum_{P_e \in A} U_i^{(e)}.$$

Recovery Phase:

Recall that as a result of the previous phases all players maintain a set A of "good" players and have shares U_i.

1. Every corrupt player $P_j \notin A$ is "rebooted".
2. Every good player $P_i \in A$ computes and sends to every corrupt player P_j the check matrix $C_{i,j} = U_i M_j^T$, i.e $C_{A,j} = U_A M_j^T$.
3. Upon receiving the data, P_j computes U_j, such that $M_i U_j^T = C_{ij}$ holds for some subset of honest, qualified players B of A, i. e. $M_B U_j^T = C_{B,j}$.
4. Player P_j sets U_j as his share.

Fig. 4. Proactive VSS—General Case [11]

in Section 4.2. Because of the observation that the symmetry of the polynomials $f^{(e)}(x,y)$ can be used to break the scheme the authors proposed two asymmetric solutions. Their first proposal is to use asymmetric polynomials instead of symmetric. The second solution is to modify the symmetric polynomial scheme by adding some "asymmetry".

Recall that as a result of the previous phases all players maintain a set A of "good" (not corrupt) players and have shares $h_i^{(t-1)}(x)$ and $g_i^{(t-1)}(y)$ polynomials of degree k (see Fig. 12). Each player P_i is associated publicly with a non-zero element $\alpha_i \in \mathbb{F}$, so $|\mathcal{P}| < |\mathbb{F}|$ (see Fig. 5).

4.4. The Second Type of Attack—the Asymmetric Case. But this modification has a flaw. Let us consider the asymmetric protocol. We can apply nearly the same attack as described in Section 4.3, but now applied to $g_i^{(t)}(y)$ instead of $h_i^{(t)}(x)$. Assume that the attacker has corrupted player P_i in some time frame, i. e. he has the shares $h_i^{(t-1)}(x)$ and $g_i^{(t-1)}(x)$. Then P_i being detected as corrupt is "rebooted". In the renewal phase his share is updated by $h_i^{(t)}(x) \longleftarrow h_i^{(t-1)}(x) + \sum_{P_e \in A} h_{e;i}^{(t-1)}(x)$ and $g_i^{(t)}(y) \longleftarrow g_i^{(t-1)}(y) + \sum_{P_e \in A} g_{e;i}^{(t-1)}(y)$. But since $h_{e;0}^{(t-1)}(x)$ is public information the attacker is able to compute $\sum_{P_e \in A} g_{e;i}^{(t-1)}(0)$, using $g_{e;i}^{(t-1)}(0) = h_{e;0}^{(t-1)}(\alpha_i)$. Hence the attacker has discovered the "true part" of P_i's new share, namely $g_i^{(t)}(0) \longleftarrow g_i^{(t-1)}(0) + \sum_{P_e \in A} h_{e;0}^{(t-1)}(x)$. The knowledge of the "true part" of the share $g_i^{(t)}(y)$ is enough for reconstructing the secret (see Remark 8.1). Therefore incrementally breaking different set of players the attacker is able to compute the secret. Note that it does not matter whether $h_{e;0}^{(t-1)}(x) = \delta_e^{(t-1)}(x,0)$ or $g_{e;0}^{(t-1)}(y) = \delta_e^{(t01)}(0,y)$ is broadcast since the attack is symmetric.

4.5. Patching the Scheme—Symmetric Case. Now we present the proposition based on symmetric polynomials given in [2] (see Fig. 6).

4.6. The Third Type of Attack. The second modification of the scheme from [2] has also a flaw. In [3] D'Arco and Stinson provided description of our attack and our fix to their scheme (see also Sect. 5.1). Now we will demonstrate our attack [12] that uses the weakness in the second proactive protocol by D'Arco and Stinson. First, note that instead of a "verification vector" $\mathbf{v^i} \in \mathbb{F}^n$ one can use a polynomial $v_i(x)$ of degree k, such that $v_i(\alpha_j) = \mathbf{v^i}_j$. In fact, we can change the last step of the renewal phase as follows: $v_i^{(t)}(x) \longleftarrow v_i^{(t-1)}(x) + \sum_{P_e \in A} h_{e;i}^{(t-1)}(x)$. In this way the size of the verification share becomes

Proactive VSS—Threshold Asymmetric Case [2, 3]

Renewal Phase:

In this phase each player P_e plays the role of the dealer.

1. Each player $P_e \in A$ selects a random polynomial $\delta_e^{(t-1)}(x, y)$ of degree k, subject to $\delta_e^{(t-1)}(0, 0) = 0$.
2. Player P_e sends $h_{e;i}^{(t-1)}(x) = \delta_e^{(t-1)}(x, \alpha_i)$ and $g_{e;i}^{(t-1)}(y) = \delta_e^{(t-1)}(\alpha_i, y)$ to P_i for $1 \leq i \leq n$ and broadcasts $h_{e;0}^{(t-1)}(x) = \delta_e^{(t-1)}(x, 0)$.

Modified Detection:

1. Player P_i checks whether $g_{e;i}^{(t-1)}(0) = h_{e;0}^{(t-1)}(\alpha_i)$ and $h_{e;0}^{(t-1)}(0) = 0$.
2. If the conditions are satisfied, then P_i computes and sends to P_j the check value $g_{e;i}^{(t-1)}(\alpha_j)$. Otherwise P_i broadcasts an accusation to P_e.
3. All players perform the usual pair-wise checking with accusations protocol and update the set of good players A.
4. All players update their shares by putting

$$h_i^{(t)}(x) \longleftarrow h_i^{(t-1)}(x) + \sum_{P_e \in A} h_{e;i}^{(t-1)}(x)$$
$$g_i^{(t)}(y) \longleftarrow g_i^{(t-1)}(y) + \sum_{P_e \in A} g_{e;i}^{(t-1)}(y).$$

Recovery Phase:

1. Every corrupt player $P_j \notin A$ is "rebooted".
2. Every good player $P_i \in A$ computes and sends to every corrupt player P_j the values $h_i^{(t-1)}(\alpha_j)$ and $g_i^{(t-1)}(\alpha_j)$.
3. Upon receiving the data, P_j computes $h_j^{(t-1)}(x)$ and $g_j^{(t-1)}(y)$, such that $h_j^{(t-1)}(\alpha_i) = g_i^{(t-1)}(\alpha_j)$, $g_j^{(t-1)}(\alpha_i) = h_i^{(t-1)}(\alpha_j)$ and $h_j^{(t-1)}(\alpha_j) = g_j^{(t-1)}(\alpha_j)$ hold for some subset of honest, qualified players $P_i \in B$ and $B \subseteq A$.
4. Player P_j takes $h_j^{(t-1)}(x)$ and $g_j^{(t-1)}(y)$ as his shares.

Fig. 5. Proactive VSS—Threshold Asymmetric Case [2, 3]

Proactive VSS—Threshold Symmetric Case [2]

Renewal Phase:

In this phase each player P_e plays the role of the dealer.

1. Each player $P_e \in A$ selects a random symmetric polynomial $\delta_e^{(t-1)}(x, y)$ (note) of degree $k-1$.
2. Player P_e sends $h_{e;i}^{(t-1)}(x) = \delta_e^{(t-1)}(x, \alpha_i)$ to P_i for $1 \leq i \leq n$.

Modified Detection:

1. Player P_i computes and sends to P_j the usual check value $h_{e;i}^{(t-1)}(\alpha_j)$.
2. All players perform the usual pair-wise checking with accusation protocol and update the set of good players A.
3. All players update their shares by putting

$$h_i^{(t)}(x) \longleftarrow h_i^{(t-1)}(x) + \alpha_i \sum_{P_e \in A} h_{e;i}^{(t-1)}(x).$$

Moreover, the players update a "verification vector" $\mathbf{v^i} \in \mathbb{F}^n$ by computing for $1 \leq j \leq n$

$$\mathbf{v^i}_j \longleftarrow \mathbf{v^i}_j + \sum_{P_e \in A} h_{e;i}^{(t-1)}(\alpha_j).$$

Detection Phase:

1. Player P_i computes and sends to player P_j the check-values $h_i^{(t-1)}(\alpha_j)$.
2. Each player P_j checks whether $h_i^{(t-1)}(\alpha_j) - h_j^{(t-1)}(\alpha_i) = \mathbf{v^j}_i \,.\, (\alpha_j - \alpha_i)$. If P_j finds that this is not true, then he broadcasts an accusation to P_i.
3. Each player P_i computes the set of good players.

Recovery Phase:

1. All corrupt players $P_j \notin A$ are "rebooted" and every good player $P_i \in A$ computes and sends to every corrupt P_j the check-values $h_i^{(t-1)}(\alpha_j)$ and $\mathbf{v^i}_j$.
2. Upon receiving the data, P_j computes $h_j^{(t-1)}(x)$, such that for some subset of honest, qualified players $P_i \in B$, $B \subseteq A$ the equation $h_j^{(t-1)}(\alpha_i) = h_i^{(t-1)}(\alpha_j) + \mathbf{v^i}_j \,.\, (\alpha_i - \alpha_j)$ holds. He calculates $\mathbf{v^j}$ by setting $\mathbf{v^j}_i = \mathbf{v^i}_j$.
3. Player P_j sets as his shares $h_j^{(t-1)}(x)$ and $\mathbf{v^j}$.

Fig. 6. Proactive VSS—Threshold Symmetric Case [2]

k. Denote the polynomial $h_{A;i}^{(t-1)}(x) = \sum_{P_e \in A} h_{e;i}^{(t-1)}(x)$. Note that the usual share $h_i^{(t)}(x)$ is "asymmetric", whereas the "symmetry" is collected in the verification share $v_i^{(t)}(x)$. But the information from the share and the verification share of a player P_i allows the attacker to calculate the initial share $h_i^0(x)$ of P_i, obtained from the Dealer during the Distribution, i.e., in the Share phase. Indeed, after q executions of the *Renewal* phase, player P_i possesses

$$h_i^{(t=q)}(x) = h_i^{(t=0)}(x) + \alpha_i \sum_{t=1}^{q} h_{A;i}^{(t)}(x) \quad \text{and} \quad v_i^{(t=q)}(x) = \sum_{t=1}^{q} h_{A;i}^{(t)}(x).$$

Assume that the attacker has corrupted P_i and has obtained $v_i^{(t=q)}(x)$ and $h_i^{(t=q)}(x)$. Subtracting $\alpha_i v_i^{(t=q)}(x)$ from $h_i^{(t=q)}(x)$ the attacker obtains the initial share $h_i^{(t=0)}(x)$. As a consequence if the adversary breaks into $k+1$ players once, even in different periods, he collects $k+1$ initial shares and hence he can recover the secret.

5. The Modified Protocol.

5.1. The Threshold Case. In this section we will describe the modifications of the protocols [12] (also [3]) that resist the attacks presented in Section 4. Let us consider the symmetric polynomial protocol in the threshold case. Basically, the problem in the procedure of D'Arco and Stinson [2] is due to the "asymmetry" in the renewal polynomial. Indeed, we have

$$f^{(t)}(x,y) \longleftarrow f^{(t-1)}(x,y) + y.\delta^{(t-1)}(x,y),$$

where $\delta^{(t-1)}(x,y) = \sum_{P_e \in A} \delta_e^{(t-1)}(x,y)$. Note that $f^{(t)}(0,0)$ is not changed, so the secret stays the same. Also $f^{(t)}(0,y)$ is changed randomly so the adversary is not able to calculate the new "true parts" of the player's shares. But to be able to perform a pair-wise check one needs a "symmetry", that is why the players keep two shares: the actual share and the verification share, which collects the asymmetry of the protocol. We propose to keep the symmetry in the renewal polynomial as follows:

$$f^{(t)}(x,y) \longleftarrow f^{(t-1)}(x,y) + (x+y).\delta^{(t-1)}(x,y).$$

Thus we need to modify only the last step in the *Renewal phase* of the Protocol in Fig. 6. All players update their shares by

$$h_i^{(t)}(x) \longleftarrow h_i^{(t-1)}(x) + (x+\alpha_i) \sum_{e \in A} h_{e;i}^{(t-1)}(x)$$

and we do not need a verification share anymore. Therefore the remaining phases can be used from the Protocol in Fig. 3 instead of the Protocol in Fig. 6.

Now we are ready to refine the conditions for security of proactive VSS (Theorem 3.1), based on the considered approach to renew player's shares by sharing 0 in the new adversary model.

Theorem 5.1. *A (b, k, n) proactive VSS computationally secure against a $(k-1, k_a)$-adversary exists if and only if $n > k + k_a$ and $k_a \leq b$. A (b, k, n) proactive VSS unconditionally secure against a $(k-1, k_a)$-adversary exists if and only if $n > 2k_a + k$ and $k_a \leq b$.*

P r o o f. The proof is identical to the VSS security proof with the only difference that the shares $h_i(x)$ and the polynomials $f(x, y)$ are of degree k instead of $k-1$. Thus in order to correct up to k_a errors (corrupt player's shares) in the Reconstruction and Recovery phase we need $n > 2k_a + k$. □

5.2. The General Case. A similar to the previous section approach [12] can be applied to the general access structure Protocol from Fig. 4, see Fig. 7.

Theorem 5.2. *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 the protocol described in Fig. 7 is a perfect proactive VSS scheme secure against $(\Delta, \Delta_{\mathcal{A}})$-adversary if the following conditions are satisfied:*

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

P r o o f. We rely again on the VSS security proof, so the third condition implies that we have a VSS secure against an $(\widetilde{\Delta}, \Delta_{\mathcal{A}})$-adversary. Consider the Recovery phase in Fig. 4. Since $B \in \Gamma(\mathcal{M})$ we have that the "true part" of P_j's share, namely $U_j\boldsymbol{\varepsilon}$, is uniquely determined by $U_j\boldsymbol{\varepsilon} = C_{B,j}^T\boldsymbol{\lambda}$, where $M_B^T\boldsymbol{\lambda} = \boldsymbol{\varepsilon}$. We know from the pair-wise checking protocol that there exists a solution U_j such that $M_B U_j^T = C_{B,j}^T$ holds, i. e., U_j is consistent with U_B. But in order to provide consistency of P_j's new share U_j with the rest of the shares U_C for $C \cap B = \emptyset$, we require U_j to be the unique solution of the system $M_B U_j^T = C_{B,j}^T$. This requirement implies the first condition in the theorem. Let us assume that there exists a group $B \in \Delta^+$, such that $\mathrm{rank}(M_B) = d-1$ holds. We know also that $\boldsymbol{\varepsilon}$ is not in $\mathrm{span}(M_B)$. Since the first row and column in $R^{(e,2)}$ are zero the players in B could solve the system $\binom{\boldsymbol{\varepsilon}^T}{M_B} R = \binom{\mathbf{0}}{U_B^{(e,2)}}$ and the solution is $R = R^{(e,2)}$.

Therefore the players in B could calculate also the "true part" of the shares $U_i^{(e)}$ for all players, since $U_i^{(e)}\boldsymbol{\varepsilon} = U_i^{(e,2)}\boldsymbol{\varepsilon}$ and $U_i^{(e,2)}$ is already revealed. That is why we require the second condition. □

Remark 5.3. In the threshold case in order to have a (b, k, n) proactive VSS secure against $(k-1, k_a)$-adversary we need $\widetilde{\Gamma} = T_{k,n}$, $k_a \leq b < k$. The first two conditions of Theorem 5.2 are fulfilled for the corresponding Vandermonde matrix.

The (b, k, n) SSSs are also called *ramp* SSSs, and they correspond to incomplete access structures. The same observation holds for general case where $(\Gamma(\mathcal{M}), \Delta)$ form incomplete access structure too. Thus we need to transform an complete (threshold or general) access structure to incomplete (ramp or general) ones in order to provide security against the mobile adversary.

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 5.4. 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.

6. 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

Proactive VSS—General Case [12]

Renewal Phase:

In this phase each player P_e plays the role of the dealer.

1. Each player $P_e \in A$ selects a random symmetric $(d-1)\times(d-1)$ matrix $R^{(e)}$ and uses it to construct two symmetric $d\times d$ matrices $R^{(e,1)}, R^{(e,2)}$. The matrix $R^{(e,1)}$ is constructed by adding a zero column and a zero row as last row and column, whereas the matrix $R^{(e,2)}$ is constructed by adding a zero column and a zero row as first row and column.
2. Player P_e sends $U_i^{(e,1)} = M_i R^{(e,1)}$ and $U_i^{(e,2)} = M_i R^{(e,2)}$ to P_i for $1 \le i \le n$, $i \neq e$.

Modified Detection:

1. Player P_i checks whether the last column of $U_i^{(e,1)}$ and the first column of $U_i^{(e,2)}$ are zero-columns.
2. If these conditions are not satisfied, P_i will broadcast an accusation to P_e. Otherwise P_i computes $U_i^{(e)}$ as the sum of the right shift of the columns of $U_i^{(e,1)}$ and the left shift of the columns of $U_i^{(e,2)}$, i. e., if we represent the matrices by columns as follows $U_i^{(e,2)} = [\mathbf{0}, (U_i^{(e,2)})_{(1)}, \ldots, (U_i^{(e,2)})_{(d-1)}]$ and $U_i^{(e,1)} = [(U_i^{(e,1)})_{(1)}, \ldots, (U_i^{(e,1)})_{(d-1)}, \mathbf{0}]$ then $U_i^{(e)} = [(U_i^{(e,2)})_{(1)}, (U_i^{(e,2)})_{(2)} + (U_i^{(e,1)})_{(1)}, \ldots, (U_i^{(e,2)})_{(d-1)} + (U_i^{(e,1)})_{(d-2)}, (U_i^{(e,1)})_{(d-1)}]$.
3. Finally, P_i computes and sends to P_j the usual check matrices $C_{ij}^{(e)} = U_i^{(e)} M_j^T$, $C_{ij}^{(e,1)} = U_i^{(e,1)} M_j^T$ and $C_{ij}^{(e,2)} = U_i^{(e,2)} M_j^T$.
4. All players perform the usual pair-wise checking with accusation protocol and they update the set of good players A.
5. All players update their shares by putting

$$U_i \longleftarrow U_i + \sum_{P_e \in A} U_i^{(e)}.$$

Fig. 7. Proactive VSS—General Case [12]

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 and form corresponding to A recombination vector) such that $\sum_{w\in A} r_w s_w = s$. Now every player P_v combines the auxiliary shares he 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. For the sake of simplicity we will present only the threshold protocols, the generalization to general access structure is straightforward using MSPs. 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.

6.1. A Simple Re-Sharing Protocol. Every player P_u holds a share $h_u^{(t-1)}(x)$. The shares are derived from a symmetric polynomial $f^{(t-1)}(x, y)$ by setting $y = \alpha_u$ (see Fig. 11). So, in the renewal phase the new shares $h_v^{(t)}(x)$ are computed.

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

A Simple Re-Sharing Protocol

Re-Sharing Phase:

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

$$\delta_{u;v}^{(t-1)}(0) = \delta_u^{(t-1)}(0, \alpha_v) = h_u^{(t-1)}(\alpha_v) = h_v^{(t-1)}(\alpha_u).$$

   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", qualified players A, 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:

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

Fig. 8. A Simple Re-Sharing Protocol

"rebooted" and in the renewal phase his share is updated. Note that $\delta_{u;v}^{(t-1)}(0) = h_v^{(t-1)}(\alpha_u)$ holds. But the attacker is able to compute $\sum_{u \in A} r_u \; \delta_{u;v}^{(t-1)}(0) = \sum_{u \in A} r_u \; h_v^{(t-1)}(\alpha_u)$. Thus he knows the "true part" of the P_v's new share, namely $h_v^{(t)}(0) = \sum_{u \in A} r_u \; \delta_{u;v}^{(t-1)}(0)$. 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.

6.2. Re-Sharing Protocol with Randomization. Another drawback of the protocol described in the previous section 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* and proposing a more efficient *commitment sharing protocol* which preserve the symmetry (see [1]).

As in the previous section we consider the following scenario see Fig. 9. Every player P_u holds a share $h_u^{(t-1)}(x)$. The shares are derived from a symmetric polynomial $f^{(t-1)}(x,y)$ by setting $y = \alpha_u$ (see Fig. 11).

Re-Sharing Protocol with Randomization

Re-Sharing Phase:

1. Each player P_u re-shares the "true part" of his share, i. e. $h_u^{(t-1)}(0)$, by choosing a symmetric polynomial $\delta_u^{(t-1)}(x,y)$ 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 $\delta_{u;v}^{(t-1)}(x)$ 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 $h_u^{(t-1)}(0)$, P_u broadcasts a k-degree polynomial $g_u^{(t-1)}(x) = h_u^{(t-1)}(x) - \delta_u^{(t-1)}(x,0)$. Note that if P_u is honest dealer then $g_u^{(t-1)}(0) = 0$ holds.
5. Each player P_v verifies that $g_u^{(t-1)}(0) = 0$ and that
$$g_u^{(t-1)}(\alpha_v) = h_u^{(t-1)}(\alpha_v) - \delta_u^{(t-1)}(\alpha_u, 0) = h_v^{(t-1)}(\alpha_u) - \delta_{u;v}^{(t-1)}(0).$$
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, qualified 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:
$$h_v^{(t)}(x) \longleftarrow \sum_{u \in A} r_u \, \delta_{u;v}^{(t-1)}(x).$$

Fig. 9. Re-Sharing Protocol with Randomization

In the same way as in the previous section 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 $\delta_{u;v}^{(t-1)}(0) = h_v^{(t-1)}(\alpha_u) - g_u^{(t-1)}(\alpha_v)$ and that $g_u^{(t-1)}(x)$ is public. Thus the attacker is able to compute $\sum_{u\in A} r_u \delta_{u;v}^{(t-1)}(0) = \sum_{u\in A} r_u \ (h_v^{(t-1)}(\alpha_u) - g_u^{(t-1)}(\alpha_v))$. He knows the "true part" of the P_v's new share, namely $h_v^{(t)}(0) = \sum_{u\in A} r_u \ \delta_{u;v}^{(t-1)}(0)$. Therefore, again incrementally breaking different sets of players the attacker is able to compute the secret.

On the negative side we do not know unconditionally secure perfect proactive VSS protocols, based on the considered approach (to re-share the player's shares). On the positive side now we can improve the conditions for security of computationally secure proactive VSS (Theorem 5.1).

Theorem 6.1. *A (k, n) threshold scheme is a proactive VSS computationally secure against a (k, k_a)-adversary exists if and only if $n > k + k_a$.*

7. Conclusions. We have shown that several unconditionally secure schemes can be broken when mobile adversary is considered, 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 in the unconditional case whether we can do better than Theorem 5.1 and Theorem 5.2, using for example the re-sharing approach instead of commitment to 0.

REFERENCES

[1] Cramer R., I. Damgard, U. Maurer. General Secure Multi-Party Computation from any Linear Secret Sharing Scheme,. EUROCRYPT'2000, LNCS 1807, Springer-Verlag, 2000, 316–334.

[2] D'Arco P., D. Stinson. On Unconditionally Secure Proactive Secret Sharing Scheme and Distributed Key Distribution Centers. Manuscript, May 2002.

[3] D'Arco P., D. Stinson. On Unconditionally Secure Robust Distributed Key Distribution Centers. ASIACRYPT'2002, LNCS 2501, Springer-Verlag, 2002, 346–363.

[4] Feldman P. A practical scheme for non-interactive verifiable secret sharing. FOCS'1987, 427–437.

[5] Frankel Y., P. Gemmell, P. MacKenzie, M. Yung. Proactive RSA. CRYPTO'1997, LNCS 1294, Springer-Verlag, 1997, 440–454.

[6] Frankel Y., P. Gemmell, P. MacKenzie, M. Yung. Optimal-resilience proactive public-key cryptosystems. FOCS'1997, 384–393.

[7] Jarecki S. Proactive Secret Sharing and Publik Key Cryptosystems. M.Sc. Thesis, 1995, MIT.

[8] Gennaro R., S. Jarecki, H. Krawczyk, T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. EUROCRYPT'1999, LNCS 1592, Springer-Verlag, 1999, 295–310.

[9] Herzberg A. S. Jarecki, H. Krawczyk, M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. CRYPTO'1995, LNCS 963, Springer-Verlag, 1995, 339–352, (extended version 1998).

[10] Herzberg A., M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung. Proactive Public Key and Signature Systems. ACM'1997—Computer and Communication Security. Springer-Verlag, 1997, 100–110.

[11] Nikov V., S. Nikova, B. Preneel, J. Vandewalle. Applying General Access Structure to Proactive Secret Sharing Schemes. Proc. Benelux, 2002, 197–206, Cryptology ePrint Archive: Report 2002/141.

[12] Nikov V., 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, 422–437.

[13] Nikov V., S. Nikova. On Proactive Secret Sharing Schemes. SAC'2004, LNCS 3357, Springer-Verlag, 2004, 314–331.

[14] Ostrovsky R., M. Yung. How to withstand mobile virus attack. PODC'1991, 51–59.

[15] Pedersen T. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. CRYPTO'1991, LNCS 547, 129–140.

[16] Stinson D., R. Wei. Unconditionally Secure Proactive Secret Sharing Scheme with combinatorial Structures. SAC'1999, LNCS 1758, Springer-Verlag, 1999, 200–214.

8. Appendix We first present Feldman's computational secure VSS protocol (see Fig. 10).

Feldman's—VSS [4]

Sharing Phase:

Let s be a secret from $\mathbb{F} = \mathbb{Z}_{\mathtt{p}}$. Let g be a randomly chosen generator of a (multiplicative) group $\mathbb{G}$ of prime order $\mathtt{q}$. Each player P_i is associated publicly with a non-zero element $\alpha_i \in \mathbb{F}$, so $|\mathcal{P}| < |\mathbb{F}|$.

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_i is computed by $\mathcal{D}$ from $s_i = f(\alpha_i)$ and then transmitted privately to player P_i.
3. Let $f(x) = \sum_{j=0}^{k} a_j x^j$. The dealer broadcasts the values (commitments) $C_j = g^{a_j}$ for $j = 0, 1, \ldots, k$.

Detection Phase:

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

Reconstruction Phase:

1. Each player P_i broadcasts s_i.
2. Take $k+1$ broadcast values for which $g^{s_i} = \prod_{j=0}^{k} C_j^{\alpha_i^j}$ holds.
3. Determine $\widetilde{f}(x)$ of degree at most k that passes through these points. Output $\widetilde{f}(0)$.

Fig. 10. Feldman's—VSS [4]

Next we present two unconditional secure VSS protocols. The first one is based on symmetric bivariate polynomials and the second protocol is based on non-symmetric bivariate polynomials (see Fig. 11 and Fig. 12).

Remark 8.1. Notice that the roles of the polynomials $h_v(x)$ and $g_v(y)$ in the protocol of Fig. 12 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 $s' = f_u(0,0)$.

VSS—symmetric polynomial

Sharing Phase:

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

1. Dealer $\mathcal{D}$ 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 $h_u(\alpha_v)$ to P_v for $1 \le v \le n$, $(v \ne u)$.
2. Each player P_v checks whether $h_v(\alpha_u) = h_u(\alpha_v)$ for $1 \le v \le n$, $(v \ne u)$. If P_v finds that this is not true, then P_v broadcasts an accusation to P_u.
3. For each player P_u, who has been accused by a qualified group of players, the dealer must broadcast his share $h_u(x)$. 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_u computes the maximum subset $A \subseteq \mathcal{P}$, such that no player in A accused another player in A. If $|A| \ge n - k_a$, then P_u accepts his share. Otherwise, P_u accuses the dealer.
5. If there are more than k_a 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 $h_v(0)$ to each $P_u \in A$.
2. After having received the $h_v(0)$'s, P_u computes a polynomial $f_u(0,y)$, such that $f_u(0,\alpha_v) = h_v(0)$ for all v with $P_v \in A$ using error-correction techniques, e. g. Reed-Solomon codes for efficiency reasons.
3. The player P_u computes and outputs $s' = f_u(0,0)$.

Fig. 11. Threshold VSS (symmetric case)

Next we present a general access structure unconditional secure VSS protocol (see Fig. 13).

VSS—asymmetric polynomial

Sharing Phase:

Let s be a secret from some finite field $\mathbb{F}$. Each player P_i is associated publicly with a non-zero element $\alpha_i \in \mathbb{F}$, so $|\mathcal{P}| < |\mathbb{F}|$.

1. Dealer $\mathcal{D}$ 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 $g_u(\alpha_v)$ to P_v for $1 \leq v \leq n$, $(v \neq u)$.
3. Each player P_v checks whether $h_v(\alpha_u) = g_u(\alpha_v)$ 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.
4. For each player P_u, who has been accused by a qualified group of players, the dealer must broadcast his shares $h_u(x)$ and $g_u(y)$. 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_u computes the maximum subset $A \subseteq \mathcal{P}$, such that no player in A accused another player in A. If $|A| \geq n - k_a$, then P_u accepts his share. Otherwise, P_u accuses the dealer.
6. If there are more than k_a 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 $h_v(0)$ to each $P_u \in A$.
2. After having received the $h_v(0)$'s, P_u computes a polynomial $f_u(0,y)$, such that $f_u(0, \alpha_v) = h_v(0)$ for all v with $P_v \in A$ using error-correction techniques for Reed-Solomon codes.
3. The player P_u computes and outputs $s' = f_u(0,0)$.

Fig. 12. Threshold VSS (asymmetric case)

VSS—general case [11]

Sharing Phase:

1. Dealer $\mathcal{D}$ chooses at random a symmetric $d \times d$ matrix R subject to s (the secret) being in its upper left corner.
2. Dealer $\mathcal{D}$ gives share $U_i = M_i R$ to player P_i (U_i is a $|\varphi(P_i)| \times d$ matrix). We call the first column of U_i (i. e. $U_i\boldsymbol{\varepsilon}$) the "true part" of the share.

Detection Phase:

1. The player P_i computes the $|\varphi(P_i)| \times |\varphi(P_j)|$ check matrix $C_{ij} = U_i M_j^T$ and, for all $j \neq i$, sends it privately to P_j.
2. Each player P_j checks for all $j \neq i$ whether $C_{ij}^T = C_{ji}$. If P_j finds that this is not true then P_j broadcasts an accusation to P_i.
3. For each player P_i, who has been accused by a qualified group of players, the dealer must broadcast his share U_i. 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 any 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_i computes the maximum subset $A \subseteq \mathcal{P}$, such that no player in A accused another player in A (i. e. is consistent). If $A^c \in \Delta_{\mathcal{A}}$, then P_i accepts his share, otherwise P_u accuses the dealer.
5. If the group of players that have broadcast an accusation to the dealer is not in $\Delta_{\mathcal{A}}$, then $\mathcal{D}$ is declared corrupt, and the protocol is stopped.

Reconstruction Phase:

1. Each player $P_i \in A$ sends U_i to each $P_j \in A$.
2. After receiving all shares of the players in A, P_j again applies the non-interactive pair-wise checking for all received matrices and computes a subset of consistent shares $B \subseteq A$ and $B \in \Gamma$.
3. Next, since $B \in \Gamma$, P_j computes a recombination vector $\boldsymbol{\lambda}$ such that $M_B^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$.
4. Finally, the secret is computed as $s = \langle \boldsymbol{\lambda}, U_B \boldsymbol{\varepsilon} \rangle$.

Fig. 13. VSS—general case [11]

Ventzislav Nikov
Innovation and Development Center Leuven
NXP Semiconductors, Belgium
e-mail: venci.nikov@gmail.com

Svetla Nikova, Bart Preneel
Department Electrical Engineering
ESAT/COSIC
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10
B-3001 Heverlee-Leuven, Belgium
e-mail: svetla.nikova
e-mail: bart.preneel@esat.kuleuven.be

Received March 30, 2007
Final Accepted September 13, 2007