DPROT Final Work

Publicly Verifiable Secret Sharing

Martí Fernández Caballé & Ismael Douha Prieto

MCYBERS - December 2020

Contents

1 Introduction

In this document we are going to analyze a scheme that allows to share a secret with many users. A part from that, and its the feature that makes it special, makes possible a verification of the shares by any entity. This means that not only the participants of the share can verify if all the process is being perform in a correct way, anybody can verify it, adding a layer of security and trust to external agents.

This scheme, as the title's says, is called **Publicly Verifiable Secret Sharing (PVSS)** and was first introduced by Berry Schoenmakers, a cryptography researcher from Eindhoven University of Technology, in 1999.

We structured this work in the following way: First, we made a brief explanation of the schemes that PVSS is based on. Second, we explained PVSS in a shallow way, not going in too much details, more in a informative way in order to give a global view of the scheme and how it works. Next, we went in more detail about PVSS and we gave a possible approach of how to do all the steps and the verification correctly. As this scheme can be applied in many context and can solve some problems, we also added a application of PVSS to our analysis. For that reason, we explained a generalization of the detailed case in order to be applicable in a real case and an application of it, e-Voting. Finally, we exposed our conclusions of the work.

2 Context

Before entering in the explanation of the scheme, we want to make a brief explanation of some schemes that are used by PVSS in order to achieve its purpose. These schemes are **Perfect Secret Sharing** and **Verifiable Secret Sharing**.

2.1 Perfect Secret Sharing

This scheme makes possible to share a secret among different (n) users. Each participant P_i receives a sub-part of the secret S_i. This distribution is made by the **Dealer**, who has the secret and wants to share it with the rest of users. After the secret is shared, only a authorized set of users can recover the secret. The main point that makes the scheme perfect is that in case an unauthorized subset of users tries to recover the secret, no information is recovered from it.

Based on this scheme, different variations can be added to be applied in real cases. One example is **Shamir Secret Sharing**, which uses random polynomials to distribute the share among the users and Lagrange polynomial interpolation to recover the secret. In case that not all the participants has the same power, **Weighted Secret Sharing** is an option. This scheme is based on Shamir's solution and gives more shares to the users with more power in order to distribute correctly the weights for the recovery phase. A generalization of Shamir's scheme is **Linear Secret Sharing**, which instead of using the public value $\varepsilon_i \in Z_q^x$, it uses vectors $v_i \in Z_q^d$.

2.2 Verifiable Secret Sharing

On top of PSS, Verifiable Secret Sharing adds a new layer of security, making possible to verify if the dealer is honest or not and in case that the dealer results dishonest, the protocol is aborted. In case that the dealer was honest, all the participants recovers the same secret.

An example of VSS is **Feldman Secret Sharing**, which using Shamir's scheme as base, the dealer publishes $A_0 = g^s$ and $A_j = g^{a_j}$. Then each participant P_i check the dealer's honesty check the equality $g^{s_i} = \prod_{j=0}^{t-1} A_j^{\varepsilon_i^j}$, where t is the number of users need to recover the secret. Another example is **Pedersen Secret Sharing**, which combines the public information of two different instances of Shamir's scheme, the dealer publishes $A_0 = g^s h^{s'}$ and $A_j = g^{a_j} h^{a'_j}$ and again the dealer's honesty is verified checking the equality $g^{s_i} h^{s'_i} = \prod_{j=0}^{t-1} A_j^{\varepsilon_i^j}$.

3 PVSS

We can consider a publicly verifiable secret sharing (PVSS) scheme, introduced by Stadler [1], a new layer on top of VSS. It is a verifiable secret sharing scheme with the property that the validity of the shares distributed by the dealer can be verified by any party and not limited to the respective participants receiving the shares.

For this reason the part where this protocol have to resist a malicious dealer sending incorrect shares to some or all participants have to be public verified. Apart from the resistance to a participants submitting incorrect shares during the reconstruction protocol that we can assume from VSS.

In general in PVSS schemes we can assume that the secret is computationally hidden, this is as the communication in this protocol should be public verifiable and use of private channels don't allow it. How public key cryptography is computationally hidden a PVSS schemes is computationally hiding.

3.1 Non-interactive model

A global non-interactive PVSS model can be the one described in [2] and [3]. We consider non-interactive because any party is able to sort out the correct shares and pool them together, this means that all *PROOFs* can be verified without iteration between players. For example in VSS you need interaction between participants to complain if the dealer is honest or not and if he received an incorrect share or not.

How we mention before this protocol no use private channels between the dealer and participants, instead it works with public key encryption through public channels authenticated using each register public key in Initialization part.

There are n participants $P_1, .., .P_n$.And a dealer D wants to share a secret value $s \in \Sigma$ among the participants requiring a subset of it to recover the secret. One option is a (t, n)-threshold scheme where we need minimum of t participants, $1 \leq t \leq n$, to reconstruct the secret without care with participants are. Also for less than t mustn't be possible to retrieve any information.

This scheme is divided in 3 protocols: Initialization, Distribution and Reconstruction.

3.1.1 Initialization

This protocol are the initial one. There are no interaction between dealer and participants. Each participant can enter or leave the system when he wants with the requirement that it hols a registered public key.

The initialization process is the one in which:

1. All system parameters are generated as part of the initialization.
2. Each participant P_i registers a public key to be used with public key encryption method E_i.

3.1.2 Distribution

In this protocol the secret is distributed by the dealer among the participants, as in VSS, and the public verification of the shares is performed. The two main steps are:

1. *Distribution of the shares*: the dealer D distributes the secret $s \in \Sigma$ among the participants. The dealer do the following steps:
   - Creates $s_1, s_2 ... s_n$ for each participant $P_1, P_2 ... P_n$ respectively.
   - Publishes the encrypted share $E_i(s_i)$ for each P_i.
   - Publishes a string $proof_D$ to show that each E_i encrypts s_i

   This $proof_D$ is the one that guarantees that the reconstruction protocol will result in the same s and it can be public verifiable.
2. *Verification of the shares*: Any party knowing public keys for the encryption methods E_i may verify the shares using the published PROOFs. In this part the verifier (any party) do:
   - Run a non-interactive verification algorithm with $PROOF_D$ to verify that $E_i(s_i)$ is a correct encryption of the share for P_i.
   - If one or more verification fails the dealer fails and the protocol is aborted.

   Using this protocol we don't need that the participants complains about correct shares because can be public verified.

   Also we can use a (t, nc)-threshold scheme, where c is the number of verifications that failed, to obtain some some level of fault-tolerance.

3.1.3 Reconstruction

This protocol are in which the secret is recovered by pooling the shares of a qualified subset of participants. The steps are:

1. *Decryption of the shares*: Each participant decrypt their shares.

   - The participant P_i decrypts their share of the secret s_i using $E_i(s_i)$
   - The participant release s_i plus a string $proof_{p_i}$ this shows the released share is correct.

   In this part we can allow fault-tolerance because it's not required that all participants succeed in decrypting $E_i(s_i)$ as long as a qualified set of participants are successful to decrypt s_i.

2. *Pooling the shares*: participants collaborate to obtain the secret.

   - Using the strings $proof_{p_i}$ to exclude the participant which are dishonest or failed to decrypt $E_i(s_i)$.
   - Reconstruction s can be done from the shares of any qualified set of participants.

3.2 Homomorphic Secret Sharing

We can asume that the PVSS have the homomorphic property. That means combining shares of independent secrets in such a way that reconstruction from the combined shares results in a combined secret.

$$E_i(s_i) \otimes E_i(s_i') = E_i(s_i \oplus s_i')$$

In case of PVSS a operation on the shares is equivalent to the encrypted shares.

4 Schoenmakers

In this document we talked about the PVSS scheme by Berry Schoenmakers [2], but this can be considered a general scheme also introduced by Standler [1] (Schoenmakers introduce the PROOFs) and Fujisaki and Okamoto but he have his specific one.

His scheme works which any group for which the discrete log problem is intractable and relates the security to Diffie-Hellman assumption and its decisional variant.

There are other schemes that but the Schoenmakers constrution [2], as he says, will be much simple than Standler or Fujisaki and Okamoto among others and achieves improvements both in efficiency and in the type of intractability assumptions. In Shoenmaker's construction only need techniques that work in any group for which the discrete log problem is intractable and rely on simple primitives. Also as author says "the performance is not only asymptotically optimal, but also good in practice".

The running time is $O(nk)$, where k is a security parameter, and n is the number of participants

4.1 Chaum-Pedersen Protocol

Before explain the special construction we introduce the Chaum and Pedersen protocol to prove: $log_{g1}h1 = log_{g2}h2$ for $g_q, h_1, g_2, h_2 \in G_q$. The prover knows a α such that $h_1 = g_1^\alpha$ and $h_2 = g_2^\alpha$, and the steps to verify are:

1. The prover chooses a random $r \in Z_q$ ($r \in_R Z_q$) and sends $a_1 = g_1^r$ and $a_2 = g_2^r$.
2. The verifier sends a random challenge $c \in_R Z_q$
3. The prover responds with $s = r - \alpha c \pmod q$
4. The verifier checks $a_1 = g_1^s h_1^c$ and $a_2 = g_2^s h_2^c$

Denote this protocol as: $DLEQ(g_1, h_1, g_2, h_2)$ Replacing the random c for a 'secure hash' function with m as input value we can use in a non-interactive way.

4.2 Special

The Schoenmakers special construction of a simple PVSS scheme was designed to share secrets chosen uniformly at random from a large set and for a (t, n)-threshold access structure, but it can be applied to any monotone access structure for which a linear

secret sharing scheme exists. We supose that G_q is a group of prime order q where the computation of discrete logarithms in this group is infeasible. Then we want that no party knows the discrete log of g with respect to G, for this reason we consider g, G generators of G_q selected independently. The dealer will achieve this by first selecting $s \in_R Z_q$ and then distributing shares of the secret $S = G^s$. This approach allows to keep the required proofs simple and efficient.

We will use the protocol by Chaum and Pedersen explained before as a subprotocol.

4.2.1 Initialization

In this protocol, like in model, we do:

1. The group G_q and the generators g, G are selected using an appropriate public procedure.
2. Participant P_i generates a private key $x_i \in_R Z_q^*$ and registers $y_i = G^{x_i}$ as its public key.

4.2.2 Distribution

This protocol consists of two steps:

1. *Distribution of the shares*:
   (a) The dealer picks a random polynomial p of degree at most $t-1$ with coefficients in Z_q.

$$p(x) = \sum_{j=0}^{t-1} \alpha_j x^j$$

   (b) Sets a $s = \alpha_0$. And keeps this polynomial secret.
   (c) Publishs:
   - The related commitments $C_j = g^{\alpha_j}$ for $0 \leq j < t$.
   - The encrypted shares $Y_i = y_i^{p(i)}$ for $1 \leq i < n$ using the public keys of the participants.

   (d) The dealer shows that the encrypted encrypted shares are consistent by producing a proof of knowledge of unique $p(i)$ for $1 \leq i \leq n$ satisfying $X_i = g^{p(i)}$ and $Y_i = y_i^{p(i)}$. This non-interactive proof is the n-fold parallel composition of the protocols for $DLEQ(g, X_i, y_i, Y_i)$. Applying Fiat-Shamir's technique, the challenge c for the protocol is computed as a cryptographic hash of X_i, Y_i, a_{1i}, a_{2i} $1 \leq i \leq n$.

The proof consists of the common challenge c and the n responses r_i.

(note: that $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$)

2. *Verification of the shares*:

   (a) The verifier computes $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$ from the C_j values.

   (b) The verifier computes a_{1i} and a_{2i} using y_i, X_i, Y_i, r_i for $1 \leq i \leq n$ and c as inputs.

$$a_{1i} = g^{r_i} X_i^c$$

$$a_{2i} = y_i^{r_i} Y_i^c$$

   (c) Checks that $H(X_i, Y_i, a_{1i}, a_{2i})$ for $1 \leq i \leq n$ matches c.

4.2.3 Reconstruction

1. *Decryption of the shares*: Each participant:

   - Using its private key x_i finds $S_i = G^{p(i)}$ from Y_i by computing $S_i = Y_i^{1/x_i}$
   - Publish S_i plus a proof that the value S_i is a correct decryption of Y_i by $DLEQ(G, y_i, S_i, Y_i)$ knowing $y_i = G^\alpha$ and $Y_i = S_i^\alpha$

2. *Pooling the shares*: The secret is obtained by a Lagrange interpolation.

$$\prod_{i=1}^{t} S_i^{\lambda_i} = \prod_{i=1}^{t} (G^{p(i)})^{\lambda_i} = G^{\sum_{i=1}^{t} p(i)\lambda_i} = G^{p(0)} = G^s$$

   where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ is a Lagrange coefficient.

To reconstruct the secret value S which is equal to G^s we only need the related values $S_i = G^{p(i)}$. Know the values of the exponents $p(i)$ are not necessary. In addition how the private key (x_i) of each participant P_i is not exposed they can use it several times in PVSS scheme. And how we mention before this scheme is homomorphic. We can see because for instance, how Schoenmakers says, given the dealer's output for secrets G^{s_1} and G^{s_2} , the combined secret $G^{s_1+s_2}$ can be obtained by applying the reconstruction protocol to the combined encrypted shares $Y_{i1}Y_{i2}$.

4.3 Security

In order to achieve the level of security desired and necessary to use this scheme in real cases, there are some assumptions that have been taken.

Lemma 1: *Under the Diffie-Hellman assumption, it is infeasible to break the encryption of the shares.*
Breaking Diffie-Hellman implies that given $x = g^{\alpha}$ and $y = g^{\beta}$, we obtain $z = g^{\alpha\beta}$ using an algorithm A. We introduce $x^{\alpha'}$, $y^{\beta'}$, g^{γ} and $y^{\beta'\gamma}$ and A outputs $z' = g^{\alpha\alpha'\beta\beta'}$ with success probability of ε. From the previous output of A, we can obtain $z'^{1/('\beta')} = g^{\alpha\beta}$ $= z$ with the same success probability ε.

Lemma 2: *Suppose that $t-1$ participants pool their shares and obtain the secret. Then we can break the Diffie-Hellman assumption.*
Again, given $x = g^{\alpha}$ and $y = g^{\beta}$, where α and β are chosen randomly, we want to obtain $z = g^{\alpha\beta}$ and to do it, we need a system that makes possible to achieve the purpose. First, we set $G = g^{\alpha}$ and $C_0 = g^{\beta} = g^{p(0)}$, since we need that the first commitment is related to the polynomial of 0. With that, we can compute the values $X_i = g^{p(i)}$ and $Y_i = g_i^{p(i)}$ for i = 1 .. $t-1$, but we can not compute the values from t to n, because $p(0)$ was not given explicitly. However, it is possible to compute the Lagrange interpolation to obtain the values $X_i = g^{p(i)}$. After that, the last step to set the system is computing the public keys (y_i) of the t to n participants, where $y_i = g_{w_i}$ for random $w_i \epsilon Z_q$, setting $Y_i = X_i^{w_i} = y_i^{p(i)}$. With that, know we can suppose that they can compute $G^{p(0)}$ from $G = g$ and $p(0) = \beta$ which results in g^{β}. If all of these things happens, then the Diffie-Hellman assumption is broken and we proved the *Lemma 2*.

Under these two lemmas that we explained and proved, the following theorems can be extracted:

Theorem 1: *Under the Diffie-Hellman assumption, the special PVSS scheme is secure in the random oracle model. That is, (i) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (ii) any non-qualified set of participants is not able to recover the secret.*

Theorem 2: *Under the DDH assumption and the random oracle assumption, the special PVSS scheme is secure. That is, (i) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (ii) any non-qualified set of participants is not able to recover any (partial) information on the secret.*

4.4 General

In section ***4.2 Special*** we explained how it works the special scheme of PVSS introduced by Berry Schoenmakers but this scheme does not allow to share whatever value that the dealer wishes, being necessary a generalization of the scheme to be applied in real scenarios.This generalization allows the dealer to share a secret $\sigma \epsilon \sum$, where $2 \leq |\sum| \leq q$ and a uniformly distribution over ϵ is not required. To do it, we have two methods.

The first method that we have, the dealer chooses a random secret $s \ \epsilon \ Z \ q$ which will act as a key and a σ which will be the value to share. Then, the dealer publishes $U = \sigma \oplus H(G^s)$ as the encrypted value, using H as a hash function. To recover the share, is enough computing $\sigma = U \oplus H(G^s)$ during the reconstruction protocol.

The second method works in the case that $\sum \subseteq G_q$. In this method, the dealer runs again the distribution protocol choosing $s \ \epsilon \ Z \ q$ as a random secret and publishes $U = \sigma(G^s)$. Similar to the previous method, the reconstruction protocol needs to compute $\sigma = U \ / \ G^s$ to recover the dealer's share.

For these methods, the reconstructed σ belongs to $\sum$ in case that $\sum = \{0,1\}^u$ and the range of the hash is also equal to $\{0,1\}^u$ for the first method and $\sum = G_q$ in the second one. For other cases, is necessary a proof of knowledge that shows $\sigma \epsilon \sum$. It is also possible that this proof is not necessary, because they decided to discard the recovered σ in case that does not belong to $\sum$.

5 E-voting

Once we know that the special scheme of PVSS can be generalized, lets see a possible application of it. In this case, the application chosen is e-Voting. Nowadays, change the way we vote is a trend since more governments, companies, political parties and so on are trying to use e-Voting instead of the classical solution, but at the end, some of these elections have particular requirements that e-Voting can not fulfill. Next we will see how PVSS can be use as e-Voting solution in small-scale elections where the voters does not need to be anonymous, which will prevent double voting.

In order to use this scheme we need a bulletin board, were the votes will be posted, a set of voters and a set of tallier (voters and talliers sets can be the same). This scheme, as you can guess from the previous sections, it is composed by two phases. A point to take into account is that now, the voters will have the role of the dealer in the previous sections and the talliers will act as a participants of each dealer's share. Before these phases, each tallier should register his/her public key in order to later be able to retrieve the votes.

During the first phase, called ***Ballot casting***, each voter runs the distribution protocol and V performs his/her vote $v\epsilon\{0,1\}$. This v value will be encrypted in the following way $U = G^{s+v}$, where $s\ \epsilon R\ Zq$. A part from that, a PROOFu will be published with the distribution protocol's output and U. This PROOFu proves that v is 0 or 1 without revealing it, making possible the public verifiable property that PVSS adds to VSS and PSS.

The second phased, named ***Tallying***, uses the reconstruction protocol explained previously but using the homomorphic property, which makes possible to anonymize the vote of each voter. Supposing that all voters has a valid ballot, each tallier computes $Y_j^* = \prod_{j=1}^{m} Y_{ij} = y_i^{\sum_{j=1}^{m} p_j(i)}$, where Y_j^* has all the accumulated encrypted shares and m represents the number of voters. After that, each tallier applies the reconstruction protocol and the homomorphic property, producing $G^{\sum_{j=1}^{m} p_j(0)} = G^{\sum_{j=1}^{m} s_j}$. Then we obtain $G^{\sum_{j=1}^{m} v_j}$ from $\prod_{j=1}^{m} U_j = G^{\sum_{j=1}^{m} s_j + v_j}$, making possible that each tallier gets the election's result computing $\sum_{j=1}^{m} v_j$.

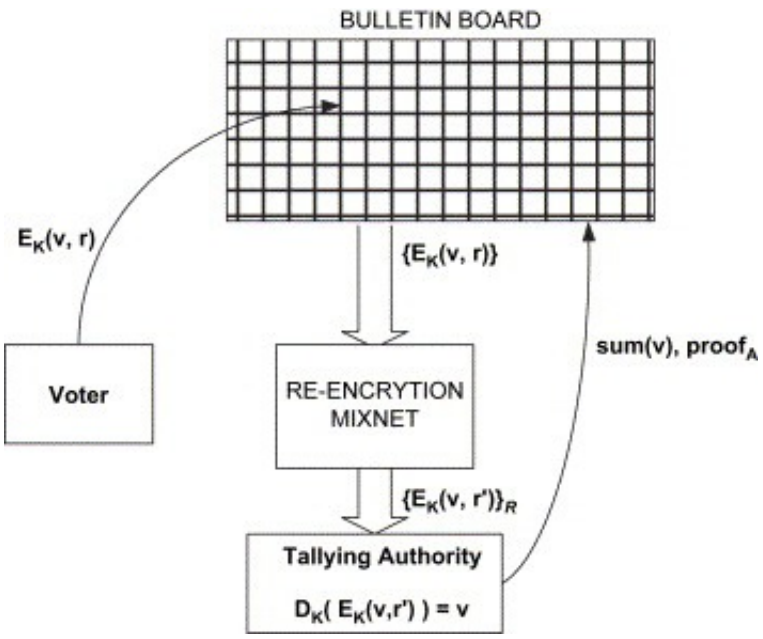

Figure 1: e-Voting scheme. *Source: ScienceDirect.*

In the previous figure, a graphical representation of how the elections are performed can be seen.

6 Conclusions

During this final work we have increased our knowledge about secret sharing, not only knowing how it works Publicly Verifiable Secret Sharing scheme in detail, how can be used and real applications of it, we also have learned more about Perfect Secret Sharing and Verifiable Secret Sharing, a part from what we learned during the lectures.

In addition to the knowledge that we have acquired in this field, we also have increased our mathematical background since both of us we come from computer science degrees and there mathematics have a secondary role, but this work and the whole course gives us more knowledge in the field and we are very happy with it.

This work also helped us with other subjects. One of the main applications of PVSS is e-Voting and during this MCYBERS we have covered this topic during CSMAGT subject, where we have done a explanation about electronic voting systems and we included a explanation about PVSS and how can be used in real elections.

7 References

[1] https://www.ubilab.org/publications/print_versions/pdf/sta96.pdf

[2] https://www.win.tue.nl/ berry/papers/crypto99.pdf

[3] https://en.wikipedia.org/wiki/Publicly_Verifiable_Secret_Sharing