Secure Multiplication of Shared Secrets In The Exponent

Rosario Gennaro* Mario Di Raimondo†

April 10, 2003

Abstract

We present a new protocol for the following task. Given tow secrets a, b shared among n players, compute the value g^{ab}.

The protocol uses the generic BGW approach for multiplication of shared secrets, but we show that if one is computing "multiplications in the exponent" the polynomial randomization step can be avoided (assuming the Decisional Diffie-Hellman Assumption holds). This results in a non-interactive and more efficient protocol.

1 Introduction

In this paper we deal with an interesting variation on a typical problem of *Secure Multiparty Computation*: the multiplication in the exponent of two secrets distributed by a scheme of *Secret Sharing*. In other words, suppose you have two (integer) secrets a, b shared among n players via secret sharing. The players want to compute the value g^{ab} (in an appropriate group) without revealing any more information about a, b beyond what the result of the computation gives out.

This problem is a variation of the well-know *multiplication of shared secrets* problem, where the parties instead want to compute the product ab or a sharing of it. This problem received a very elegant solution in one of the seminal papers in Secure Multiparty Computation [3], for secrets shared using the Shamir secret sharing scheme [11].

Let us recall the BGW solution [3]. Remember that in Shamir's secret sharing, the sharing of a secret a is achieved as follows. Each player P_i holds a value $a_i = f(i)$ where f is a random t-degree polynomial subject to the condition that $f_a(0) = a$. It is very simple to see that a coalition of t players has no information about the value a. Now assume that each player P_i holds two shares $a_i = f_a(i)$ and $b_i = f_b(i)$ for secrets a, b respectively. The value $c_i = a_i \cdot b_i = f(i)$ where $f = f_a \cdot f_b$. So the c_i's seem to constitute a sharing of the value $c = ab$. Unfortunately there are two problems: (1) the polynomial f is of degree $2t$; (2) the polynomial f is not random: in particular it's the product of two polynomials of degree t.

The first problem can be tolerated if there are enough players in the network. For example if $n > 3t + 1$ and the bad players are at most t, then it is still possible (in principle) for the good players to still reconstruct $c = ab$. But, if this sharing of c is used repeatedly, this could lead to an increase of the degree of the sharing polynomial which could prevent the good players from carrying out the required computation. For this reason in [3] a degree reduction step is implemented to bring the degree of the sharing polynomial back to t.

*IBM T.J.Watson Research Center. `rosario@watson.ibm.com`

†Dipartimento di Matematica e Informatica, Università di Catania, Italy. `diraimondo@dmi.unict.it`

The second problem is even more subtle. The fact that f is not random, could lead to unwanted release of information. For example, if all the players want to do is compute c, they should be able to reveal the shares c_i in their possession and find the free term of f by polynomial interpolation. But this process reveals the whole polynomial f, which could then be factored as $f = f_a \cdot f_b$ and the individual secrets a and b would be exposed. For this reason a randomization step is included in the [3] solution which transforms the f polynomial into a random one with the same free term.

Our solution. In our case we do not want to compute $c = ab$ but rather g^c.

In order to keep things simple, we assume $n > 3t+1$ and do not implement the degree-reduction step since the sharing of c will be used only once and the degree of any sharing polynomial will never go above $2t$.

We focus instead on the second problem. The value c can be computed via polynomial interpolation from $2t + 1$ correct shares c_i. As it is well known, polynomial interpolation is a linear transformation, so this means that we can write

$$c = \sum_{i \in T} \lambda_{i,T} c_i$$

for any set T of $2t + 1$ indices, and publicly known Lagrangian coefficients $\lambda_{i,T}$.

Thus we can ask each player to reveal $\gamma_i = g^{c_i}$ and compute

$$g^c = \prod_{i \in T} \gamma_i^{\lambda_{i,T}}$$

Notice that we do not perform any randomization step. The intuition is that because the actual shares are "masked" inside the exponent of g, it should be hard to get any extra information about the polynomial f.

We prove that this intuition is correct. If the *Decisional Diffie-Hellman (DDH) Assumption* holds, we can prove that the above protocol is secure, i.e. it reveals only the value g^c and all the rest of the adversary's view can be simulated. We recall that the DDH Assumption says that given three values $A = g^a, B = g^b, C = g^c$, it is hard to decide if $c = ab$ or random. In our protocol we are dealing with a similar situation: given the values $\gamma_i = g^{c_i}$ we would like to claim that it is hard to decide if the c_i are points on a random polynomial of degree $2t$ or points on a polynomial of degree $2t$ that can be factored as the product of two polynomials of degree t.

Of course the above description is very informal and in particular it glosses over the problems caused by malicious players who may not perform the protocol as instructed. In particular the important question is how to recognize correct shares γ_i. We can tolerate such an active adversary and details are shown in the rest of the paper.

The main improvement is the gain in efficiency. By removing the randomization step, the protocol becomes completely non-interactive (if there are no faults). Beyond this substantial round complexity improvement, the protocol becomes also computationally less intensive for the players.

1.1 Related Work

We already mentioned the BGW protocol [3] as the starting point for our contribution. We actually use a somewhat simplified version of it from [7]. The robust solution in which the adversary can be malicious uses ideas from the protocol of [1].

2 Preliminaries

Number Theory. In the following we denote with p, q two prime numbers such that $q|(p-1)$. We consider the subgroup G_q of Z_p^* of order q and let g be a generator for G_q. All computations are mod$p$ unless otherwise noted.

We are going to assume that the *Decisional Diffie-Hellman Assumption* (DDH) holds in G_q, i.e. no probabilistic polynomial time algorithm given as input three values (g^a, g^b, g^c) can decide if $c = ab \bmod q$ with probability better than 1/2. For a discussion on the DDH see [4].

A formal definition of the DDH follows. Let $PRIMES(n)$ be the set of pairs (p, q) where both p, q are prime numbers, $q|p-1$, $|q| = n$ and $|p| = poly(n)$ for some polynomial $poly(\cdot)$. Choose (p, q) at random in $PRIMES(n)$ and g as a random element of order q in Z_p^*. Given an algorithm $\mathcal{A}$ we can define the following probabilities

$$PR_{\mathcal{A},R}(n) = \mathsf{Prob}_{a,b,c \in_R Z_q}[\mathcal{A}(p, q, g, g^a, g^b, g^c) = 1]$$

$$PR_{\mathcal{A},DH}(n) = \mathsf{Prob}_{a,b \in_R Z_q}[\mathcal{A}(p, q, g, g^a, g^b, g^{ab}) = 1]$$

We say that the Decisional Diffie-Hellman Assumption holds if, for any probabilistic polynomial time algorithm $\mathcal{A}$ and for any polynomial $P(\cdot)$, there exists and index n_P such that for all $n > n_P$ we have

$$|PR_{\mathcal{A},R}(n) - PR_{\mathcal{A},DH}(n)| < \frac{1}{P(n)}$$

Communication Model. We assume that there are n players $\mathcal{P}_1, ..., \mathcal{P}_n$. They are connected by a complete network of private (i.e. untappable) point-to-point channels. In addition, the players have access to a dedicated broadcast channel; by dedicated we mean that if player $\mathcal{P}_i$ broadcasts a message, it is received by every other player and recognized as coming from $\mathcal{P}_i$. These assumptions (privacy of the communication channels and dedication of the broadcast channel) allow us to focus on a high-level description of the protocols. However, it is worth noting that these abstractions can be substituted with standard cryptographic techniques for privacy, commitment and authentication.

We assume that the communication channels between the players provide a *partially synchronous* message delivery. That is we assume that the messages sent during the protocol, are received by their recipients within some fixed time bound. Notice that this will allow a malicious adversary to wait for the messages of the honest players in a given round before sending her own. In the cryptographic protocols literature this is also known as a *rushing* adversary.

The Adversary. As already seen, the adversary can corrupt at most t of n players, with $n > 3t$. Moreover she is *active*, i.e. she can alter the behavior of the corrupted players to deviate them from the enstablished protocol. For example, the corrupted players can send spurious messages to gain additional informations about the shared secrets.

Secure Multiparty Computation. In the model above, the task of Secure Multiparty Computation can be described as follows. Each party P_i has a private input x_i, and there is a public function f over n inputs. The players want to compute $y = f(x_1, \ldots, x_n)$ without exposing any other information about their private inputs. Secure multiparty computation was introduced in [8, 12].

The informal notion of "not revealing any more information" can be formalized as follows (using definitions proposed in [2, 9, 5]). We can imagine a trusted (i.e. not corruptible by the adversary) party which receives all the inputs x_i and returns the value y. In this case it's clear that the adversary learns nothing else beyond the value y. In the real execution the adversary instead has a *view* which is a random variable including all the messages seen by the adversary, together with its internal randomness and the inputs of the corrupted players.

We say that a protocol is secure, if this view can be *simulated*. I.e. if there exists a simulator which runs in the ideal world with the trusted party, which is able to produce a view with a probability distribution which is indistinguishable from the real one.

3 VSS Protocols

Here we recall a few existing techniques that we use in our solution.

SHAMIR'S SECRET SHARING. In Shamir's secret sharing protocol [11], a dealer shares a secret among n players $\mathcal{P}_1, \ldots, \mathcal{P}_n$ in the following way. Given a number $t < n$, a prime q and a secret $s \in Z_q$, the dealer chooses at random a polynomial $S(\cdot)$ over Z_q of degree t, such that $S(0) = s$. It then secretly transmits to each player $\mathcal{P}_i$ a share $s_i = S(i) \bmod q$. The secret can easily be reconstructed by polynomial interpolation by at least $t+1$ players. It is not hard to see that a collection of t players has no information about the secret. Notice also that this protocol does not tolerate a malicious adversary (a bad dealer could give shares that do not lie on a polynomial of degree t and/or bad players may prevent reconstruction by giving out bad shares).

We will use the following notation to indicate the above protocol:

$$\mathsf{Shamir\text{-}SS}[s] \xrightarrow[t,n]{S} [s_i]$$

FELDMAN'S VSS. Feldman's Verifiable Secret Sharing (VSS) protocol [6], extends Shamir's secret sharing method in a way to tolerate a malicious adversary which corrupts up to $\frac{n-1}{2}$ players *including the dealer*.

Like in Shamir's scheme, the dealer generates a random t-degree polynomial $S(\cdot)$ over Z_q, s.t. $S(0) = s$, and transmits to each player $\mathcal{P}_i$ a share $s_i = S(i) \bmod q$. The dealer also broadcasts values $V\!s_k = g^{a_k}$ where a_k is the k^{th} coefficient of $S(\cdot)$. This will allow the players to check that the values s_i really define a secret by checking that

$$g^{s_i} = \prod_k (V\!s_k)^{i^k} \bmod p \tag{1}$$

If the above equation is not satisfied, player $\mathcal{P}_i$ asks the dealer to reveal his share (we call this a *complaint*). If more than t players complain then the dealer is clearly bad and he is disqualified. Otherwise he reveals the share s_i matching Equation (1) for each complaining $\mathcal{P}_i$.

Equation (1) also allows detection of incorrect shares s'_i at reconstruction time. Notice that the value of the secret is only computationally secure, e.g., the value $g^{a_0} = g^s$ is leaked. However, it can be shown that an adversary that learns t or less shares cannot obtain any information on the secret s beyond what can be derived from g^s. This is good enough for some applications, but it is important to notice that it does not offer "semantic security" for the value s. I.e. the adversary can verify, using the revealed informations, if a given $\tilde{s}$ is the shared secret.

We will use the following notation to denote the execution of a Feldman's VSS protocol.

$$\mathsf{Feldman\text{-}VSS}[s](g) \xrightarrow[t,n]{S} [s_i](V\!s_k)$$

PEDERSEN'S VSS. We now recall a VSS protocol that provides information theoretic secrecy for the shared secret. This is in contrast to Feldman's VSS protocol which leaks the value of g^s. The protocol is due to Pedersen [10].

Pedersen's VSS uses the parameters p, q, g as defined for Feldman's VSS. In addition, it uses an element $h \in Z_p^*$ such that h belongs to the subgroup generated by g and the discrete log of h in base g is unknown (and assumed hard to compute).

The dealer first chooses two t-degree polynomials $S(\cdot), \tilde{S}(\cdot)$, with random coefficients over Z_q, subject to $S(0) = s$, the secret. The dealers sends to each player $\mathcal{P}_i$ the values $s_i = S(i) \bmod q$ and $\tilde{s}_i = \tilde{S}(i) \bmod q$. The dealer then commits to each coefficient of the polynomials S and $\tilde{S}$ by publishing the values $V\!S_k = g^{a_k} h^{b_k}$, where a_k (resp. b_k) is the k^{th} coefficient of S (resp. $\tilde{S}$).

This allows the players to verify the received shares by checking that

$$g^{s_i} h^{\tilde{s}_i} = \prod_k (V\!S_k)^{i^k} \bmod p \tag{2}$$

As in Feldman's VSS the players who hold shares that do not satisfy the above equation broadcast a complaint. If more than t players complain the dealer is disqualified. Otherwise the dealer broadcasts the values s_i and $\tilde{s}_i$ matching the above equation for each complaining player $\mathcal{P}_i$.

At reconstruction time the players are required to reveal both s_i and $\tilde{s}_i$ and Equation (2) is used to validate the shares. Indeed in order to have an incorrect share σ_i' accepted at reconstruction time, it can be shown that player $\mathcal{P}_i$ has to compute the discrete log of h in base g.

Notice that the value of the secret is unconditionally protected since the only value revealed is $V\!S_0 = g^s h^{b_0}$ (it can be seen that for any value s' there is exactly one value b_0' such that $V\!S_0 = g^{\sigma'} h^{b_0'}$ and thus $V\!S_0$ gives no information on s).

We will use the following notation to denote an execution of Pedersen's VSS:

$$\textsf{Pedersen-VSS}[s, \tilde{s}](g, h) \xrightarrow[t,n]{S,\tilde{S}} [s_i, \tilde{s}_i](V\!S_k)$$

3.1 The protocol

Assume that you have two secrets r, s shared via a Shamir's secret sharing with t-degree polynomials R, S (let's limit ourselves to the case of a passive adversary for simplicity). The players want to publicly compute the product rs without revealing any information about r, s.

As pointed out in [3], the seminal paper on multiparty computation, it is not secure for each player to broadcast the product share $\sigma_i = r_i \cdot s_i$. Given $2t+1$ product shares σ_i we can interpolate rs, but unfortunately we can compute a lot more than that. Indeed the interpolation yields the product polynomial $R \cdot S$, which can be easily factored producing the component secrets r, s. [3] suggests to solve this problem by adding a random polynomial of degree $2t$ and null free term, before the interpolation.

Now assume that instead of computing rs, the players want to compute g^{rs}. Is it still necessary to perform this polynomial randomization step? We show that if we assume the DDH Assumption, then it is sufficient for each player to reveal the value $g^{r_i s_i}$. Given a set Λ of $2t+1$ of these values, one can compute g^{rs} by simple interpolation in the exponent.

$$g^{sr} = \prod_{i \in \Lambda} (g^{s_i r_i})^{\lambda_{i,\Lambda}} \bmod p$$

via the appropriate Lagrangian coefficients $\lambda_{i,\Lambda}$.

The intuition behind the proof is that under the DDH assumption it is impossible for the adversary to gain any information about the single shares r_i and s_i "contained" in the value $g^{s_i r_i}$. Below we present a full simulation argument.

This proof yields a substantial efficiency improvement in protocols that require the multiplication of shared secrets in the exponent.

If we need to deal with a malicious adversary then we assume that r, s are shared via a Pedersen's VSS and combine our new idea with the multiplication protocol by Abe [1]. This will allow us to obtain a robust protocol without resorting to expensive zero-knowledge proofs. The final result is presented in Figure 1.

Mult-Exp

Private Input for player $\mathcal{P}_i$: $(s_i, \tilde{s}_i), (r_i, \tilde{r}_i)$
Public Input: $V\!S_k, V\!R_k$ for $k \in [0..t]$

1. Let

$$E\!S_i = \prod_{k=0}^{t} {V\!S_k}^{i^k} \bmod p \text{ and } E\!R_i = \prod_{k=0}^{t} {V\!R_k}^{i^k} \bmod p$$

then player $\mathcal{P}_i$ acts as a dealer in the following VSS protocols:

$$\mathsf{Pedersen\text{-}VSS}[s_i, \tilde{s}_i](g,h) \overset{S_i,\tilde{S}_i}{\underset{t,n}{\longrightarrow}} [\alpha_{ij}, \tilde{\alpha}_{ij}](\langle E\!S_i\rangle V\!S_{i1} \dots V\!S_{it})$$

$$\mathsf{Pedersen\text{-}VSS}[r_i, \tilde{r}_i](g,h) \overset{R_i,\tilde{R}_i}{\underset{t,n}{\longrightarrow}} [\beta_{ij}, \tilde{\beta}_{ij}](\langle E\!R_i\rangle V\!R_{i1} \dots V\!R_{it})$$

$$\mathsf{Feldman\text{-}VSS}[s_i \cdot r_i](g) \overset{S_i \cdot R_i}{\underset{2t,n}{\longrightarrow}} [\langle \alpha_{ij} \cdot \beta_{ij}\rangle](V\!S\!R_{i0} \dots V\!S\!R_{i,2t})$$

Note: the angle brackets $\langle\cdot\rangle$ denote information which is *not* given to the other players by the dealer, since they already have it from before.
Each player $\mathcal{P}_j$ will check the behavior of $\mathcal{P}_i$ as follows:

$$g^{\alpha_{ij}} h^{\tilde{\alpha}_{ij}} \stackrel{?}{=} E\!S_i \prod_{k=1}^{t} {V\!S_{ik}}^{j^k} \text{ and } g^{\beta_{ij}} h^{\tilde{\beta}_{ij}} \stackrel{?}{=} E\!R_i \prod_{k=1}^{t} {V\!R_{ik}}^{j^k} \text{ and } g^{\alpha_{ij}\beta_{ij}} \stackrel{?}{=} \prod_{k=0}^{2t} {V\!S\!R_{ik}}^{j^k}$$

If any of this checks fail, then $\mathcal{P}_j$ broadcasts a complain towards $\mathcal{P}_i$.
For any $\mathcal{P}_j$ that complains against him, player $\mathcal{P}_i$ will broadcast the private values he sent to $\mathcal{P}_j$. If these values also fail the checks, then $\mathcal{P}_i$ is disqualified, otherwise these new values will be taken as the private information of $\mathcal{P}_j$.

2. Every player selects a group of $2t+1$ non-disqualified players. Let Λ be this set (does not have to be the same for each player). The value g^{sr} is computed as

$$g^{sr} = \prod_{i\in\Lambda} {V\!S\!R_{i0}}^{\lambda_{i,\Lambda}} = \prod_{i\in\Lambda} (g^{s_i r_i})^{\lambda_{i,\Lambda}} \bmod p \tag{3}$$

Figure 1: Protocol to compute distributively g^{rs}

Here we give an informal explanation of the protocol. Our goal is to make sure that each player $\mathcal{P}_i$ reveals the correct value $g^{s_i r_i}$. Once we are sure of that, we can compute the output by interpolation as we said above. A possible solution is to force $\mathcal{P}_i$ to prove in zero-knowledge that the value is correct, using as reference the commitments $V\!R, V\!S$. But ZK proofs are expensive and increase the amount of interaction required by the protocol. Our solution, using ideas from [1], does without them.

First notice that the values $E\!S_i$ and $E\!R_i$ are really the commitments $g^{s_i} h^{\tilde{s}_i}$ and $g^{r_i} h^{\tilde{r}_i}$ to the shares held by player $\mathcal{P}_i$ (that's because we basically evaluated "in the exponent" the commitments to the sharing polynomials of r, s). These commitments are publicly known to everybody so we ask player $\mathcal{P}_i$ to share s_i, r_i via two Pedersen-VSS using them as part of the verification information. This will assure us that he is really sharing s_i, r_i.

Then we will ask $\mathcal{P}_i$ to share $r_i \cdot s_i$ via a Feldman-VSS. This will reveal $g^{s_i r_i}$ and allow us to do the interpolation in the exponent. Here to make sure that $\mathcal{P}_i$ is acting properly we will ask him to reveal only the public information of the Feldman-VSS but to send *no* shares to the other players. Indeed the polynomial used to share $r_i \cdot s_i$ will be the product of the polynomials used in the previous two Pedersen-VSS to share r_i and s_i. This will guarantee that the value $g^{r_i s_i}$ is correct. As we will see in the proof, this step also requires the DDH assumption to make sure we are not leaking information (that's because we are performing a secret sharing with a non-random polynomial).

Theorem 1 *If $n > 3t$ and the DDH Assumption holds, then the protocol* Mult-Exp*, is a secure protocol which on input the shares of secrets r, s (distributed with two* Pedersen-VSS*), computes g^{rs}.*

4 Proof of Security for Mult-Exp

We want to show that Mult-Exp does not leak any more information to an adversary, beyond the value g^{rs}. We construct a simulator that on input the shares of the corrupted players, the public input of the protocol, and the final output g^{rs} will interact with the adversary in a simulated execution of the protocol. We need to show that this execution is indistinguishable to the adversary. But we will not be able to prove that the simulated view is identical to the real one. Rather we will prove that it is *computationally indistinguishable*, i.e. "looks identical" to a computationally bounded adversary. This last step will be proven via a reduction to the DDH Assumption.

We assume $n > 3t$ and that the players corrupted by the adversary are the first t ones: $\mathcal{P}_1, \ldots, \mathcal{P}_t$. Thus the simulator will control the remaining $n - t$ players. The simulator is described in detail in Figure 2.

Basically the simulator chooses random sharings for the honest players under the condition that the final result must "hit" g^{rs}. However in doing so, the simulator is not able to get a view which is identical to the real one.

The first difference is in the distributions of the values VSR_{i0}. In the real protocol we have that $VSR_{i0} = g^{S(i)R(i)}$ where S, R are two random polynomials of degree t. In the simulation instead $VSR_{i0} = g^{T(i)}$ where T is a $2t$-degree polynomial which follows a distribution which is statistically close to the uniform one.

The other difference is in the simulation of the Feldman-VSS of the honest players. In the real execution the values VSR_{ik} (for $k \in [0..2t]$) are of the form $g^{k-\text{coeff}(S_i \cdot R_i)}$, where $k-\text{coeff}(\cdot)$ denotes the k^{th} coefficient of a polynomial and S_i, R_i are two random polynomials of degree t. On the other hand in the simulated transcript the values VSR_{ik} are of the form $g^{k-\text{coeff}(T_i)}$, where again T_i is a $2t$-degree polynomial which follows a distribution which is statistically close to the uniform one.

We show below (Lemmas 4 and 3) that if an adversary could detect the differences shown above (i.e. distinguish between the real and the simulated execution) then she could solve the DDH problem. Thus under the DDH Assumption the simulated view is computationally indistinguishable from the real one.

Before proving the two main Lemmas, we prove a preliminary Lemma which extends the DDH Assumption in a *matrix* form. Consider the following two distributions:

$$\mathcal{MDH}(t) = [g^{a_0}, g^{a_1}, \ldots, g^{a_t}, g^{b_0}, g^{b_1}, \ldots, g^{b_t}, (g^{a_i b_j})_{i,j \in [0..t]}]_{a_i, b_j \in_R Z_q}$$

Sim-Mult-Exp

Input: $g^{sr}, (s_i, \tilde{s}_i), (r_i, \tilde{r}_i)$ for $i \in [1..t]$. Also the public values V_{S_k}, V_{R_k} for $k \in [0..t]$

1. The simulator computes the values:

$$E_{S_i} = \prod_{k=0}^{t} {V_{S_k}}^{i^k} \bmod p \text{ and } E_{R_i} = \prod_{k=0}^{t} {V_{R_k}}^{i^k} \bmod p \text{ for } i \in [1..n]$$

and also

$$V_{SR_{j0}} = g^{s_j r_j} \bmod p \text{ for } j \in [1..t]$$

The simulator now chooses $V_{SR_{i0}} \in_R G_q$ for $i \in [t+1..2t]$. The values $V_{SR_{i0}}$ determine a $2t$-degree polynomial $T(\cdot)$ in the sense that $V_{SR_{i0}} = g^{T(i)}$ and $g^{rs} = g^{T(0)}$. The simulator needs to compute the values $V_{SR_{i0}}$ for the remaining good players ($i \in [2t+1..n]$) such that it still holds that $V_{SR_{i0}} = g^{T(i)}$. This can be done again by interpolation in the exponent.

For each $j \in [2t+1..n]$ we denote with Q_j the set $\{1, \ldots, 2t\} \cup \{j\}$. Then the simulator computes:

$$V_{SR_{j0}} = \left(\frac{g^{sr}}{\prod_{i \in Q_j \setminus \{j\}} {V_{SR_{i0}}}^{\lambda_{i,Q_j}}} \right)^{(\lambda_{j,Q_j})^{-1}} \bmod p$$

where λ_{i,Q_j} are the appropriate Lagrangian coefficients.

For every honest player $\mathcal{P}_i$ ($i \in [t+1..n]$), the simulator simulates the three VSS protocols as follows:

(a) chooses $\alpha_{ij}, \tilde{\alpha}_{ij} \in_R Z_q$ and sends it to the bad player $\mathcal{P}_j$ for $j \in [1..t]$. Then he must compute the public verification information $V_{S_{ik}}$ for $k \in [1..t]$ so that it satisfies the following equation

$$g^{\alpha_{ij}} h^{\tilde{\alpha}_{ij}} = E_{S_i} \prod_{k=1}^{t} {V_{S_{ik}}}^{j^k} \text{ for } j = 1, \ldots, t$$

and this can be done again via interpolation in the exponent.

(b) chooses $\beta_{ij}, \tilde{\beta}_{ij} \in_R Z_q$ and sends it to the bad player $\mathcal{P}_j$ for $j \in [1..t]$. Then he must compute the public verification information $V_{R_{ik}}$ for $k \in [1..t]$ so that it satisfies the following equation

$$g^{\beta_{ij}} h^{\tilde{\beta}_{ij}} = E_{R_i} \prod_{k=1}^{t} {V_{R_{ik}}}^{j^k} \text{ for } j = 1, \ldots, t$$

and this can be done again via interpolation in the exponent.

(c) Performs one last interpolation in the exponent to compute the verification information $V_{SR_{ik}}$ for $k \in [1..2t]$ of the Feldman-VSS. These values must satisfy the equation

$$g^{\alpha_{ij}\beta_{ij}} = \prod_{k=0}^{2t} V_{SR_{ik}}^{j^k}$$

2. The simulator will now carry on the rest of the protocol since he has enough information to deal with complaints from the bad players.

Figure 2: The simulator for the protocol Mult-Exp

and

$$\mathcal{MR}(t) = [g^{a_0}, g^{a_1}, \ldots, g^{a_t}, g^{b_0}, g^{b_1}, \ldots, g^{b_t}, (g^{c_{ij}})_{i,j\in[0..t]}]_{a_i,b_j,c_{ij}\in_R Z_q}$$

Lemma 2 *If the DDH Assumption is true, then the distributions $\mathcal{MDH}(t)$ and $\mathcal{MR}(t)$ are computationally indistinguishable (for any value of t polynomial in the security parameter).*

PROOF:The proof follows a basic hybrid argument. For each $k = 0, 1, \ldots, (t+1)^2$ define the following hybrid distribution:

$$\mathcal{M}_k(t) = [g^{a_0}, g^{a_1}, \ldots, g^{a_t}, g^{b_0}, g^{b_1}, \ldots, g^{b_t}, (V_{ij})_{i,j\in[0..t]}]$$

where $a_i, b_j \in_R Z_q$ and $V_{ij} = g^{a_i b_j}$ if $i + (t+1)j \geq k$, or $V_{ij} \in_R G$ otherwise. Clearly $\mathcal{M}_0(t) = \mathcal{MDH}(t)$, while $\mathcal{M}_{(t+1)^2}(t) = \mathcal{MR}(t)$.

Now assume by contradiction that $\mathcal{MDH}(t)$ and $\mathcal{MR}(t)$ are not computationally indistinguishable. Then by a basic hybrid argument there exist an index k and a polytime distinguisher $\mathcal{D}$ between $\mathcal{M}_k(t)$ and $\mathcal{M}_{k+1}(t)$.

We now show how to use $\mathcal{D}$ to break the DDH Assumption. Let $(A = g^a, B = g^b, C = g^c)$ the instance of the DDH problem that we want to solve. Find the unique pair $\alpha, \beta \in [0..t]$ such that $k + 1 = \alpha + (t+1)\beta$. Then choose $a_0, \ldots, a_{\alpha-1}, a_{\alpha+1}, \ldots, a_t \in_R Z_q$. Similarly choose $b_0, \ldots, b_{\beta-1}, b_{\beta+1}, \ldots, b_t \in_R Z_q$.

Now set $g^{a_\alpha} = g^a$, $g^{b_\beta} = g^b$ and the remaining g^{a_i}, g^{b_j} accordingly. Set $V_{ij} \in_R G$ for all the indices i, j such that $i + (t+1)j \leq k$. Set $V_{\alpha,\beta} = g^c$. Finally set $V_{ij} = g^{a_i b_j}$ for all the indices i, j such that $i + (t+1) > k + 1$ (notice that we can compute this value since we know at least one of a_i or b_j).

Clearly if $g^c = g^{ab}$ this will be an instance of the distribution $\mathcal{M}_k(t)$, while if $g^c \in_R G$ we have an instance of $\mathcal{M}_{k+1}(t)$. So we can use $\mathcal{D}$ to distinguish between the two cases. □

Now we can prove the main Lemmas about the simulatability of Mult-Exp.

Lemma 3 *Let $P(\cdot), Q(\cdot)$ be random t-degree polynomials over Z_q. Let $R(\cdot)$ be a random $2t$-degree polynomial over Z_q. If the DDH Assumption is true, then the following distributions over G_q^{2t+1}*

$$\mathcal{PQ} = \{g^{k-\text{coeff}(P\cdot Q)}\}_{k=0..2t}$$

$$\mathcal{R} = \{g^{k-\text{coeff}(R)}\}_{k=0..2t}$$

are computationally indistinguishable.

PROOF:This is a simple consequence of the previous Lemma. Indeed assume you have a distinguisher $\mathcal{D}$ between $\mathcal{PQ}$ and $\mathcal{R}$. It can be easily adapted to distinguish between $\mathcal{MDH}(t)$ and $\mathcal{MR}(t)$ as follows.

Given an instance $I = [g^{a_i}, g^{b_j}, V_{ij}]$ where $V_{ij} = g^{a_i b_j}$ or random in G. Clearly you can set $g^{a_i} = g^{i-\text{coeff}(P)}$ and $g^{b_j} = g^{j-\text{coeff}(Q)}$. Now set

$$C_k = \prod_{i,j:i+j=k} V_{ij}$$

Clearly if I is an instance of $\mathcal{MDH}(t)$, we have that $C_k = g^{k-\text{coeff}(P\cdot Q)}$. Also if I is an instance of $\mathcal{MR}(t)$, we have that $C_k = g^{k-\text{coeff}(R)}$ for a random polynomial of degree $2t$. □

Lemma 4 *Let $P(\cdot), Q(\cdot)$ be random t-degree polynomials over Z_q. Let $R(\cdot)$ be a random $2t$-degree polynomial over Z_q. Let $n > 3t$ (but still polynomial in t and the security parameters, i.e. the size of q). If the DDH Assumption is true, then the following distributions over G_q^n*

$$\mathcal{PQ} = \{g^{(P(i)Q(i))}\}_{i=1..n}$$

$$\mathcal{R} = \{g^{R(i)}\}_{i=1..n}$$

are computationally indistinguishable.

PROOF:This Lemma follows from the previous one. Once you have constructed the values C_k, you can easily evaluate the n points in the exponent as

$$\Gamma_i = \prod_{k=0}^{2t} C_k^{i^k}$$

and depending on the case this would be an instance of the distribution $\mathcal{PQ}$ or of the distribution $\mathcal{R}$. $\square$

References

[1] M. Abe, Robust distributed multiplication without interaction, *Advances in Cryptology – proceedings of CRYPTO '99, Lecture Notes in Computer Science volume 1666*, Springer-Verlag, pp. 130–147, 1999.

[2] D. Beaver. Foundations of Secure Interactive Computing. *CRYPTO'91*, LNCS 576.

[3] M. Ben-Or, S. Goldwasser and A. Wigderson, Completeness Theorems for Non-cryptographic Fault-Tolerant Distributed Computations, *Proceedings of the 20^{th} Annual Symposium on the Theory of Computing*, ACM Press, pp. 1–10, 1988.

[4] D. Boneh, The Decision Diffie-Hellman Problem, *Algorithmic Number Theory – proceedings of Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science volume 1423*, Springer-Verlag, pp. 48–63, 1998.

[5] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology* 13(1):143-202, 2000.

[6] P. Feldman, A Practical Scheme for Non-Interactive Verifiable Secret Sharing, *Proceedings of the 28^{th} IEEE Symposium on Foundation of Computer Science (FOCS)*, IEEE, pp. 427–437, 1987.

[7] R. Gennaro, M.O. Rabin, T. Rabin. Simplified VSS and Fact-Track Multiparty Computations with Applications to Threshold Cryptography. *PODC 1998*. pp.101-111.

[8] O. Goldreich, S. Micali and A. Wigderson. How to play any mental game. *STOC'87*, pp.218-229.

[9] S. Micali and P. Rogaway. Secure Computation. *CRYPTO'91*, LNCS 576.

[10] T. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, *Advances in Cryptology – proceedings of CRYPTO '91, Lecture Notes in Computer Science volume 576*, Springer-Verlag, pp. 129–140, 1991.

[11] A. Shamir, How to Share a Secret, *Communications of the ACM*, vol. 22, n. 11, pp. 612–613, 1979.

[12] A. Yao. Protocols for Secure Computation. *FOCS'82*, pp.160–164.