On Distributed Key-Generation for El-Gamal Encryption

Ivan Damgård

CPT 2002

1 Introduction

In this note we give a way to generate keys for the El Gamal cryptosystem that can be executed by several players in such a way that afterwards, the players share the secret key, but no minority of them has enough information to compute the key. Furthermore, we establish a situation such that any honest majority of the players can afterwards use the secret key to decrypt a ciphertext.

2 The Goal

The paper on electronic voting by Cramer, Gennaro and Schoenmakers (CGS) starts from the following scenario: we have authorities $A_1, ..., A_n$, and a large group G_q of prime order q generated by element g (for instance a subgroup of Z_p^*, where q divides $p-1$). We let $t = \lfloor (n-1)/2 \rfloor$, i.e., t is maximal such that $t < n/2$.

Furthermore $h = g^s$ is publically known, where h is the public El-Gamal key. Each authority A_i knows a private value $s_i \in Z_q$, where $s_i = f(i)$ and where $f()$ is a random polynomial of degree at most t chosen such that $f(0) = s$. Finally, for each A_i, the value $h_i = g^{s_i}$ is publically known.

It is explained in CGS how the A_i's can use these pieces of public and private information to securely decrypt ciphertexts assuming that at most t of them will misbehave. Here ,securely means that if the honest players agree to execute a decryption, this will always succeed and no other information than the desired plaintext will be released. Furthermore, since decryption requires participation of at least $t+1$ players, no ciphertext can be decrypted by the dishonest players alone.

This scenario can be set up by assuming that a trusted party does it once and for all by simply choosing s and $f()$ at random and compute the s_i, h_i's and send s_i to A_i in private. This may be acceptable in some cases, but nevertheless creates (for a limited amount of time) a single entity that must be trusted completely, since the trusted party knows the secret key. Here, we look at a way to have the players create the keys from scratch.

3 A Key Generation Protocol

We note that the proposal we give here is not complete in the sense that more steps and complications in the protocol would be needed to construct that could be formally proved to be "as good" as when a single trusted party generates the keys. What we show here is only intended as an example to demonstrate some basic ideas. We will assume that the t dishonest players may try to misbehave and do as much damage as possible by sharing their information and coordinating their actions. An easy equivalent way to think of this is to assume a single adversary who has corrupted up to t of the players, has seen all their information and now control completely their actions.

The first basic step is the following:

- Each A_i chooses a random polynomial $f_i()$ of degree at most t, $f_i(x) = v_i + a_{i1}x + ... + a_{it}x^t$. (we give the degree 0 coefficient a special name v_i as it plays a special role in the following). He publishes $g^{v_i}, ..., g^{a_{it}}$. He sends in private $f_i(j)$ to each A_j, $j = 1..n$.

The values $g^{v_i}, g^{a_{i1}} ..., g^{a_{it}}$ serve as a sort of commitment to the coefficients of $f_i()$. Note that $g^{v_i} = g^{f_i(0)}$. Note also that having seen these values, everyone can compute $g^{f_i(j)}$ for any i, j, as follows:

$$g^{f_i(j)} = g^{v_i + a_{i1}j + ... + a_{it}j^t} = g^{v_i} \cdot (g^{a_{i1}})^j \cdots (g^{a_{it}})^{j^t}$$

Therefore, we can do the next step:

- Each A_j looks at the value u_i he received from A_i (which hopefully equals $f_i(j)$). He computes $g^{f_i(j)}$ from the public information as above and compares this to g^{u_i}. If there is a mismatch, he posts a public complaint, and now A_i must make $f_i(j)$ public. If this value is correct, A_j uses this value in the following, otherwise A_i has obviously misbehaved and is disqualified.

It may seem strange that the protocol may require a value to be made public that should otherwise be secret. But the point is the following: if A_j complains in the above step, then clearly either A_i or A_j (or both) is corrupt. If A_j is corrupt, then the adversary already knows $f_i(j)$ from

Step 1 and and it doesn't hurt to make it public. If A_i is corrupt, the adversary has chosen $f_i(j)$ itself, and again he learns nothing new.

What we have obtained is that now every plyer A_j knows correct values $f_i(j)$ for all i, except those that have been disqualified and can now be ignored. For simplicity, we will assume that no one was disqualified. Then we do the following

- Each A_j computes $s_j = f_1(j) + f_2(j) + ... + f_n(j)$. We set the public key to be $h = g^{f_1(0)} \cdot g^{f_2(0)} \cdots g^{f_n(0)} = g^{v_1+...+v_n}$. And we define $h_j = g^{f_1(j)} \cdots g^{f_n(j)}$. Note that everyone can compute h and the h_j's from the information made public earlier.

Define the polynomial $f()$ by $f() = f_1() + ... + f_n()$, and $s = f(0)$, then $s = f_1(0)+...+f_n(0) = v_1+...+v_n$, and in general $f(j) = f_1(j)+...+f_n(j)$. We then have the situation we wanted: we have a public key h and a corresponding secret key s such that $h = g^s$. We have a polynomial $f()$ such that $f(0) = s$, and each A_j knows $s_j = f_1(j) + f_2(j) + ... + f_n(j) = f(j)$. Finally we have $h_j = g^{f_1(j)} \cdots g^{f_n(j)} = g^{f(j)}$.

How secure is this protocol? we argued underway that it is guaranteed to produce values s_j, g^{s_j} for each A_j that are *correct*, i.e., there really is a polynomial $f()$ of degree at most t that is consistent with all these values. Therefore, decryption using this sharing of the secret key is guaranteed to work correctly. The remaining question is: how much can the adversary learn from the protocol about the secret key? If he can find out too much, encryption using the public key we created may not be secure. In the following, we argue that the protocol is good enough in practice in this respect, but that it cannot be formally proved to be as good as in the case where single trusted party chooses the keys. Reaching this goal requires more complex constructions. The analysis below of these questions is not mandatory reading, but is included for those interested.

3.1 How secret is the secret key?

Consider the situation where an honest A_i executes Step 1, and suppose the adversary has corrupted/brokin into a subset consisting of t A_j's. The adversary sees $g^{v_i}, g^{a_{i1}}, ..., g^{a_{it}}$ plus the values $\{f_i(j)|\ j \in S\}$. Suppose for a minute that we also knew v_i. Then it is a consequence of Lagrange interpolation that one can compute any of the coefficients a_{il} of $f_i()$ by a linear combination, i.e., there exist ν_0 and $\{\nu_j|\ j \in S\}$ such that

$$a_{il} = \nu_0 v_i + \sum_{j \in S} \nu_j f_i(j)$$

The adversary can of course not do this computation, since he doesn't know $v_i = f_i(0)$, but he *can* compute ν_0 and the ν_j's. This means that knowing g^{v_i} and $f_i(j), j \in S$ he can in fact compute the following:

$$g^{a_{il}} = (g^{v_i})^{\nu_0} \cdot \prod_{j \in S} (g^{f_i(j)})^{\nu_j}$$

Therefore, once the adversary has seen g^{v_i} and $\{f_i(j)|\ j \in S\}$, it makes no difference for him whether any of the $g^{a_{il}}$ are made public, since he could compute them himself anyway. Furthermore, since $f_i()$ has been randomly chosen of degree at most t, the values $\{f_i(j)|\ j \in S\}$ are uniformly random and independent values in Z_q. This is because there are only t of them, and it is a basic property of Shamir's secret sharing scheme that as long as only t shares are given, these contain no information about the secret (the value of the polynomial in 0), so in particular, they are independent of g^{v_i}. Hence, it makes no difference from the adversary's point of view whether the values $\{f_i(j)|\ j \in S\}$ are made public by A_i or not. He might as well choose t random values himself and this would result in a distribution perfectly indistinguishable from what he sees in real life.

This means that from the point of view of what the adversary learns, we can replace Step 1 by

– Each A_i chooses a random polynomial $f_i()$ of degree at most t, $f_i(x) = v_i + a_{i1}x + ... + a_{it}x^t$. He publishes g^{v_i}.

We conclude that the way our protocol chooses the public and secret key is equivalent to the following:

– Each honest player A_i chooses at random a value v_i and publishes g^{v_i}. Having seen these values, the adversary must choose values v_j, g^{v_j} on behalf of the corrupted players (he may also choose to have some of them disqualified). Finally, the secret key becomes $s = v_1 + ...v_n$ and the public key is $h = g^{v_1} \cdots g^{v_n}$.

The goal of the adversary is of course to choose his own values v_j in a malicious way such that the sum s becomes easier for him to guess. Of course, if he knew the contributions v_i from the honest players, he could force s to be whatever value he wanted. But in fact the v_i's are random and he only knows the g^{v_i}'s. If discrete log is a hard problem (and we must of course assume this anyway), the adversary cannot compute any of the v_i's and it is therefore reasonable to assume that he cannot choose the v_j's such that they depend on the v_i's in a way that would be useful for him. So it would not be unreasonable to use the protocol in practice.

However, since g^{v_i} does determine v_i uniquely, we *cannot* prove that the v_j's are indpendent of the v_i's and so it is not clear that s will be a uniformly random value - as it would have been if a trusted party had chosen s. Solving this technical problem requires addition of more steps to the protocol. For details on this, see the paper by Gennaro et al. from EuroCrypt 2000: *Secure Distributed Key Generation for Discrete Log based Cryptosystems*, Springer Verlag LNCS series 1592.