A Simple Attack on ElGamal Public Key Encryption

(Extended Abstract)

Dan Boneh*
dabo@cs.stanford.edu

Abstract

We present a simple attack on the ElGamal public key system. The attack applies when encryption is done in a subgroup of $\mathbb{Z}_p^*$.

1 Introduction

In its simplest form, the ElGamal system [2] encrypts messages in $\mathbb{Z}_p^*$ for some prime p. Let g be an element of $\mathbb{Z}_p^*$ of order q. The private key is a number in the range $1 \leq x < q$. The public key is a tuple $\langle p, g, y \rangle$ where $y = g^x \bmod p$. To encrypt a message $M \in \mathbb{Z}_p$ the original scheme works as follows: (1) pick a random r in the range $1 \leq x < q$, and (2) compute $u = M \cdot y^r \bmod p$ and $v = g^r \bmod p$. The resulting ciphertext is the pair $\langle u, v \rangle$.

To speed up the encryption process one often uses an element g of order much smaller than p. For example, p may be 1024 bits long while q is only 512 bits long. In the extreme one might take q to be only a 160 bits.

We note that public key systems in general, and the ElGamal system in particular, are mostly used for key management. For example, in the case of E-mail one encrypts the mail using a symmetric session-key and then encrypts the session-key using the recipient's ElGamal key. Session-keys are typically short, e.g. 128 bits. In countries with domestic and export controls session-keys are typically as short as 64 bits.

We show that *naive* ElGamal encryption of a session-key in a subgroup results in a total break. That is, an attacker can recover the plaintext of a given ciphertext using only the public key. Hence, the combination of (1) encryption in a subgroup, and (2) encryption of short messages, should be done with care.

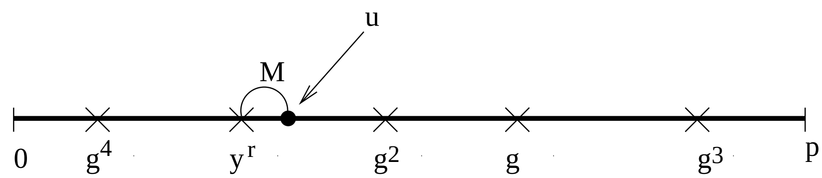
2 The subgroup rounding problems

From here on we assume $g \in \mathbb{Z}_p^*$ is an element of order q where $q \ll p$. For concreteness one may think of p as 1024 bits long and q as 512 bits long. Let G_q be the subgroup of $\mathbb{Z}_p^*$ generated by g. Observe that G_q is extremely sparse in $\mathbb{Z}_p^*$. Only one in 2^{512} elements belongs to G_q. We also assume M is a short message of length much smaller than $\log_2(p/q)$. For example, M is a 64 bits long session-key.

To understand the intuition behind the attack it is beneficial to consider a slight modification of the ElGamal scheme. After the random r is chosen one encrypts a message M by computing

*Supported by NSF.

$u = M + y^r \bmod p$. That is, we "blind" the message by *adding* y^r rather the multiplying by it. The ciphertext is then $\langle u, v\rangle$ where v is defined as before. Clearly y^r is a random element of G_q. We obtain the following picture:



The $\times$ marks represent elements in G_q. Since M is a relatively small number, encryption of M amounts to picking a random element in G_q and then slightly moving away from it. Assuming the elements of G_q are uniformly distributed in $\mathbb{Z}_p^*$ the average gap between elements of G_q is much larger than M. Hence, with high probability, there is a unique element $z \in G_q$ that is sufficiently close to u. More precisely, with high probability there will be a unique element $z \in G_q$ satisfying $|u - z| < 2^{64}$. If we could find z given u we could recover M. Hence, we obtain the additive version of the subgroup rounding problem:

Additive subgroup rounding: let z be an element of G_q and Δ an integer satisfying $\Delta < 2^m$. Given $u = z + \Delta \bmod p$ find z. When m is sufficiently small, z is uniquely determined (with high probability assuming G_q is uniformly distributed in $\mathbb{Z}_p$).

Going back to the original multiplicative ElGamal scheme we obtain the multiplicative subgroup rounding problem.

Multiplicative subgroup rounding: let z be an element of G_q and Δ an integer satisfying $\Delta < 2^m$. Given $u = z \cdot \Delta \bmod p$ find z. When m is sufficiently small z, is uniquely determined (with high probability assuming G_q is uniformly distributed in $\mathbb{Z}_p$).

An efficient solution to either problem would imply that the corresponding *naive* ElGamal encryption scheme is insecure. We are interested in solutions that run in time $O(\sqrt{\Delta})$ or, even better, $O(\log \Delta)$. In the next section we show a solution to the multiplicative subgroup rounding problem.

The reason we refer to these schemes as "naive ElGamal" is that messages are encrypted *as is*. Our attacks show the danger of using the system in this way. For proper security one must pre-format the message prior to encryption or modify the encryption mechanism. For example, one could use DHAES [1].

3 An algorithm for multiplicative subgroup rounding

We are given an element $u \in \mathbb{Z}_p$ of the form $u = z \cdot \Delta \bmod p$ where z is a random element of G_q and $|\Delta| < 2^m$. Our goal is to find Δ. As usual, we assume that m, the length of the message being encrypted, is much smaller than $\log_2(p/q)$. Then with high probability Δ is unique. For example, take p to be 1024 bits long, q to be 512 bits long and m to be 64.

Suppose Δ can be written as $\Delta = \Delta_1 \cdot \Delta_2$ where both Δ_1 and Δ_2 are $m/2$ bits each. We show how to find Δ from u in time $O(2^{m/2})$. Observe that

$$u = z \cdot \Delta = z \cdot \Delta_1 \cdot \Delta_2 \pmod p$$

Dividing by Δ_1 and raising both sides to the power of q yields:

$$(u/\Delta_1)^q = z^q \cdot \Delta_2^q = \Delta_2^q \pmod{p}$$

We can now build a table of size $2^{m/2}$ containing the values $\Delta_2^q \bmod p$ for all $\Delta_2 = 0, \ldots, 2^{m/2}$. Then for each $\Delta_1 = 0, \ldots, 2^{m/2}$ we check whether $u^q/\Delta_1^q \bmod p$ is present in the table. If so, then $\Delta = \Delta_1 \cdot \Delta_2$ is a candidate value for Δ. Assuming Δ is unique there will only be one such candidate.

The algorithm above requires $2^{m/2+1}$ modular exponentiations and $O(2^{m/2})$ space. Hence, when the system is used to encrypt a 64 bit session key, the algorithm requires on the order of eight billion exponentiations. Far less than the time to compute discrete log in $\mathbb{Z}_p^*$.

Note that the attack works only when Δ factors into a product of two integers, each approximately $m/2$ bits long. These factors need not be prime. When $m = 64$ the density of such Δ is approximately 8% (this is a heuristic estimate). Hence, roughly one out of 12 messages can be decrypted using the algorithm.

4 Summary and open problems

We showed that one should use care when encrypting short sessions-keys using the ElGamal system in a subgroup of $\mathbb{Z}_p^*$. In particular, the naive approach of encrypting messages "as is" is insecure. We presented a simple algorithm that frequently decrypts m bit messages in time $O(2^{m/2})$. When applied to 64 bit session-keys the algorithm breaks the system much faster than the time required to compute discrete log.

There are a number of open problems regarding this attack:

Problem 1: Is there a $O(2^{m/2})$ time algorithm for the multiplicative subgroup rounding problem that works for all Δ?

Problem 2: Is there a $O(2^{m/2})$ time algorithm for the additive subgroup rounding problem?

Problem 3: Can either the multiplicative or additive problems be solved in time less that $O(2^{m/2})$? Is there a sub-exponential algorithm (in 2^m)?

Acknowledgments

We thank Paul van Oorschot for several conversations on this problem.

References

[1] M. Abdalla, M. Bellare, P. Rogoway, "DHAES: An encryption scheme based on the Diffie-Hellman problem", manuscript.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on the discrete logarithm", IEEE Transactions on Information Theory, 31(4):469–472, 1985.