A correction to "Efficient and Secure Comparison for On-Line Auctions"

Ivan Damgård, Martin Geisler, and Mikkel Krøigaard

Dept. of Computer Science, University of Aarhus

Abstract. In this note, we describe a correction to the cryptosystem proposed in [1, 2]. Although the correction is small and does not affect the performance of the protocols from [1, 2], it is necessary as the cryptosystem is not secure without it.

1 Introduction

In [1, 2], the authors of this note proposed a protocol for secure comparison of integers. A homomorphic cryptosystem was also proposed upon which the protocol is based. The cryptosystem works as follows: To generate keys on input security parameters k, t, a k-bit RSA modulus $n = pq$ is chosen, along with a small ℓ-bit prime u and a t-bit prime v. It is required that both u and v divide $p - 1$ and $q - 1$. Finally, elements $g, h \in \mathbb{Z}_n^*$ are chosen, such that h has order v modulo both p and q, and g has order uv modulo both p and q. The public key is $pk = (n, g, h, u)$, the secret key is $sk = (p, q, v)$.

To encrypt a number $m \in \mathbb{Z}_u$, choose r as a random $2t$-bit integer and let the ciphertext be

$$E_{pk}(m, r) = g^m h^r \bmod n$$

It would be natural to choose the randomizer r uniformly in $\mathbb{Z}_v$, but this cannot be done since v is secret, instead it is chosen to be much larger than v since then $h^r \bmod n$ has distribution statistically indistinguishable from uniform in the group generated by h.

To decrypt ciphertext $c = E_{pk}(m, r)$, one first computes $c^v \bmod n = g^{vm} \bmod n$. The protocols from [1, 2] only requires checking if $m = 0$, so one can just verify whether $c^v \bmod n = 1$. For a small u, all of m can be recovered by just building a table of containing values of $g^{vm} \bmod n$ and the corresponding m.

Define G to be group generated by g, H to be the group generated by h. It is shown in [1, 2] that this system is semantically secure under

Conjecture 1. For any constant ℓ and appropriate choice of t as a function of k, the tuple (n, g, h, u, x) is computationally indistinguishable from (n, g, h, u, y), where (n, g, h, u) are generated as in the above key generation, x is uniform in G and y is uniform in H.

Various attacks against the assumption are studied in [1, 2], and the conclusion is that if k is large enough to make factoring infeasible and t large enough so that $2^{t/2}$ modular exponentiations are infeasible, the assumption is believed to hold.

Unfortunately, it was overlooked in [1, 2] that if v is chosen to divide both $p-1$ and $q-1$, then v also divides the (public) $n-1$. This means that if one first computes $a = (n-1)/u^j$ where u^j is the maximal power of u that divides $n-1$, then raising a number to power a outputs 1 if the number was in H and something different from 1 otherwise. Thus, the assumption is false for the construction from [1, 2].

Note that while it may seem natural to "solve" the problem by having v divide only $p-1$, say, this would not be secure either: if h still has to have order v, this would force h to be 1 modulo q, and so one could factor n just by computing $\gcd(n, h-1)$.

2 The Correction

The correction is very simple, and consists of choosing in the key generation two t-bit primes v_p, v_q, and constructing p, q such that $v_p \mid (p-1)$ and $v_q \mid (q-1)$. Then we choose g to be of order uv_pv_q and h to be of order v_pv_q. The public key is $pk = (n, g, h, u)$, the secret key is $sk = (p, q, v_p, v_q)$. The encryption and decryption are as before, except that one raises to exponent v_pv_q to decrypt, and in the encryption, the randomizer r should be chosen somewhat longer than $2t$ bits (see more details on this below).

The system is easily seen to be semantically secure under exactly the same assumption as before, however, we can now hope that the assumption is true for the new way to generate n, g, h: there is no longer an easy connection between $n-1$ and the secret key, so the attack described earlier no longer works. The study of other attacks as done in [1, 2] is still valid for the corrected system.

As in [1, 2], we recommend choosing k between 1000 and 2000 and $t = 160$.

The original as well as the corrected system are closely related to the schemes proposed by Groth in [3]. The main difference is that we specifically go for a small plaintext space defined by a small prime u that divides both $p-1$ and $q-1$. This means that our system leaves a large factor in $p-1$ and $q-1$ free to be chosen at random. This makes key generation simpler and may make n harder to factor. A more practical difference is that we can decrypt faster by computing only modulo p or q: indeed m is uniquely determined from $c^{v_p} \bmod p = g^{v_p m} \bmod p$.

A final word on performance of encryption: if we want to make sure that $h^r \bmod n$ is uniform in the group generated by h, we should choose r somewhat longer than $2t$ bits, say of length $2.5t$ bits – since in the corrected system, the order of h is $2t$ bits long. This will cause the encryption to take about 25% more time compared to the original system. However, if one is willing to make the additional assumption that raising h to a $2t$-bit exponent produces an element that is computationally indistinguishable from uniform in H, then one can keep the original encryption algorithm and this means that using the corrected system in the protocols from [1, 2] will produce exactly the same performance as reported there.

3 Acknowledgement

Thanks to Claudio Orlandi for helping us realize the problem with the original scheme.

Bibliography

[1] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007.

[2] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Homomorphic encryption and secure comparison. International Journal of Applied Cryptography vol. 1 no. 1, 2008.

[3] Jens Groth. Cryptography in subgroups of $\mathbb{Z}_n^*$. In Joe Kilian, editor, *TCC '05*, volume 3378 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2005.