Generalized Diffie-Hellman Modulo a Composite is not Weaker than Factoring

Preliminary Version

Eli Biham* Dan Boneh† Omer Reingold‡

Abstract

The Diffie-Hellman key-exchange protocol may naturally be extended to $k > 2$ parties. This gives rise to the *generalized* Diffie-Hellman assumption (GDH-Assumption). Naor and Reingold have recently shown an efficient construction of pseudo-random functions and reduced the security of their construction to the GDH-Assumption. In this note, we prove that breaking this assumption modulo a composite would imply an efficient algorithm for factorization. Therefore, the security of both the key-exchange protocol and the pseudo-random functions can be reduced to factoring.

1 Introduction

The Generalized Diffie-Hellman (GDH) Assumption was originally considered in the context of a key-exchange protocol for $k > 2$ parties (see e.g., [5, 6]). This protocol is an extension of the (extremely influential) Diffie-Hellman key-exchange protocol [1]. Given a group G and an element $g \in G$, the high-level structure of the protocol is as follows: Party $i \in [k] \stackrel{\text{def}}{=} \{1, 2, \ldots, k\}$ chooses a secret value, a_i. The parties exchange messages of the form $g^{\prod_{i \in I} a_i}$ for several proper subsets, $I \subsetneq [k]$. Given these messages, each of the parties can compute $g^{\prod_{i \in [k]} a_i}$ and this value defines their common key (in some publicly known way). Since the parties use an insecure (though authenticated) channel, it is essential that the messages they exchange do not reveal $g^{\prod_{i \in [k]} a_i}$. The GDH-Assumption is even stronger: informally, it states that it is hard to compute $g^{\prod_{i \in [k]} a_i}$ from an algorithm that can query $g^{\prod_{i \in I} a_i}$ for *any* proper subset, $I \subsetneq [k]$ of its choice. The precise statement of the assumption is given in Section 2.1.

*Computer Science Department, Technion, Haifa 32000, Israel. E-mail: biham@cs.technion.ac.il

†Dept. of Computer Science, Stanford University, Staford, CA, 94305-9045. E-mail: dabo@cs.stanford.edu

‡Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by a Clore Scholars award and by a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: reingold@wisdom.weizmann.ac.il.

Recently, another application to the GDH-Assumption was proposed by Naor and Reingold [3]. They showed an attractive construction of pseudo-random functions such that its security can be reduced to the GDH-Assumption. Motivated by this application, we provide in this note a proof that the GDH-Assumption modulo a Blum-integer is not stronger than the assumption that factoring Blum-integers is hard. Similar reductions were previously described in the context of the standard Diffie-Hellman assumption by McCurley [2] and Shmuely [5]. In fact, Shmuely [5] also provided a related proof that the GDH-Assumption (modulo a composite) itself can be reduced to factoring. Her reduction works when the algorithm that breaks the GDH-Assumption succeeds in computing $g^{\prod_{i \in [k]} a_i}$ for *every* choice of values $\langle a_1, a_2, \ldots, a_k \rangle$ (which is not sufficient for the applications of the GDH-Assumption). In contrast, our reduction works even when the algorithm breaking the GDH-Assumption only succeeds for some non-negligible fraction of the $\langle a_1, \ldots, a_k \rangle$.

2 The Assumptions

In this section we define the GDH-Assumption in $\mathbb{Z}_N^*$ (the multiplicative group modulo N), where N is a Blum-integer. We also define the assumption that factoring Blum-integers is hard. The restriction to Blum-integers is quite standard and it makes the reduction of the GDH-Assumption to factoring much simpler. In order to keep our result general, we let N (in both assumptions) be generated by *some* polynomial-time algorithm FIG (where FIG stands for factoring-instance-generator):

Definition 2.1 (FIG) *The factoring-instance-generator, FIG, is a probabilistic polynomial-time algorithm such that on input 1^n its output, $N = P \cdot Q$, is distributed over $2n - bit$ integers, where P and Q are two $n - bit$ primes and $P = Q = 3 \bmod 4$ (such N is known as a Blum-integer).*

A natural way to define FIG is to let $FIG(1^n)$ be uniformly distributed over $2n - bit$ Blum-integers but other choices were previously considered (e.g., letting P and Q obey some "safety" conditions).

2.1 The GDH-Assumption

To formalize the GDH-Assumption (which is described in the introduction) we use the following two definitions:

Definition 2.2 *Let N be any possible output of $FIG(1^n)$, let g be any quadratic-residue in $\mathbb{Z}_N^*$ and let $\vec{a} = \langle a_1, a_2, \ldots, a_k \rangle$ be any sequence of k elements of $[N]$. Define the function $h_{N,g,\vec{a}}$ with domain $\{0,1\}^k$ such that for any input, $x = x_1 x_2 \cdots x_k$,*

$$h_{N,g,\vec{a}}(x) \stackrel{\text{def}}{=} g^{\prod_{x_i=1} a_i} \bmod N$$

Define $h^r_{N,g,\vec{a}}$ to be the restriction of $h_{N,g,\vec{a}}$ to the set of all k-bit strings except 1^k (i.e., the restriction of $h_{N,g,\vec{a}}$ to $\{0,1\}^k \setminus \{1^k\}$).

Definition 2.3 (ϵ-solving the GDH$_k$-Problem) *Let $\mathcal{A}$ be a probabilistic oracle machine, $k = k(n)$ an integer-valued function and $\epsilon = \epsilon(n)$ a real-valued function. $\mathcal{A}$ ϵ-solves the GDH$_k$-Problem if for infinitely many n's*

$$\Pr[\mathcal{A}^{h^r_{N,g,\vec{a}}}(N,g) = h_{N,g,\vec{a}}(1^n)] > \epsilon(n)$$

where the probability is taken over the random bits of $\mathcal{A}$, the choice of N according to the distribution $FIG(1^n)$, the choice of g uniformly at random in the set of quadratic-residues in $\mathbb{Z}_N^*$ and the choice of each of the values in $\vec{a} = \langle a_1, a_2, \ldots, a_{k(n)} \rangle$ uniformly at random in $[N]$.

Informally, the GDH-Assumption is that there is no "efficient" oracle machine $\mathcal{A}$ that ϵ-solves the GDH$_k$-Problem for "non-negligible" ϵ. We formalize this in the standard way of interpreting "efficient" as "probabilistic polynomial-time" and "non-negligible" as "larger than $1/poly$". However, our reduction (Theorem 3.1) is more quantitative.

Assumption 2.1 (The GDH-Assumption Modulo a Blum-Integer) *Let $\mathcal{A}$ be any probabilistic polynomial-time oracle machine and $k = k(n)$ any integer-valued function that is bounded by a polynomial (and is efficiently-constructible). There is no positive constant α such that $\mathcal{A}$ $\frac{1}{n^\alpha}$-solves the GDH$_k$-Problem.*

2.2 The Factoring-Assumption

We formalize the assumption that factoring Blum-integers is hard in an analogous way:

Definition 2.4 (ϵ-factoring) *Let $\mathcal{A}$ be a probabilistic Turing-machine and $\epsilon = \epsilon(n)$ a real-valued function. $\mathcal{A}$ ϵ-factors if for infinitely many n's*

$$\Pr[\mathcal{A}(P \cdot Q) \in \{P, Q\}] > \epsilon(n)$$

where, the distribution of $N = P \cdot Q$ is $FIG(1^n)$.

Assumption 2.2 (Factoring Blum-Integers) *Let $\mathcal{A}$ be any probabilistic polynomial-time oracle machine. There is no positive constant α such that $\mathcal{A}$ $\frac{1}{n^\alpha}$-factors.*

3 Reducing the GDH-Assumption to Factoring

Theorem 3.1 *Assumption 2.1 (the GDH-Assumption) is implied by Assumption 2.2 (Factoring). Furthermore, assume that $\mathcal{A}$ is a probabilistic oracle machine with running-time $t = t(n)$ such that $\mathcal{A}$ ϵ-solves the GDH$_k$-Problem (where $k = k(n)$, is an integer-valued function that is efficiently-constructible and $\epsilon = \epsilon(n)$ a real-valued function). Then, there exists a probabilistic Turing-machine $\mathcal{A}'$ with running time $t'(n) = poly(n, k(n)) \cdot t(n)$ that ϵ'-factors, where $\epsilon'(n) = \epsilon(n)/2 - O(k(n) \cdot 2^{-n})$.*

As an intuition to the proof, assume that $\mathcal{A}$ computes $h_{N,g,\vec{a}}(1^k)$ for *any* sequence $\vec{a} = \langle a_1, a_2, \ldots, a_{k(n)} \rangle$. The algorithm $\mathcal{A}'$ can use $\mathcal{A}$ to extract square-roots in $\mathbb{Z}_N^*$ and consequently $\mathcal{A}$ can factor N (as shown in [4]). This is done as follows: $\mathcal{A}'$ first samples v

uniformly at random in $\mathbb{Z}_N^*$ and computes $g = v^{2^k} \bmod N$. For every $0 < i < k$, we have that $g^{2^{-i}} = v^{2^{k-i}} \bmod N$. Let $\vec{a} = \langle a_1, \ldots, a_k \rangle$ be the vector where for all i, $a_i = \frac{1}{2} \bmod \ell$ (here ℓ is the order of g in $\mathbb{Z}_N^*$). It follows that algorithm $\mathcal{A}'$ can easily compute $h^r_{N,g,\vec{a}}(x)$ for any $x \neq 1^k$. Hence, $\mathcal{A}'$ can invoke $\mathcal{A}$ with input $\langle N, g \rangle$ and answer every query, q, of $\mathcal{A}$ with $h^r_{N,g,\vec{a}}(q)$. Eventually, $\mathcal{A}$ outputs the value $u = h_{N,g,\vec{a}}(1^n) = g^{2^{-k}} \bmod N$. We now have that $u^2 = v^2 \bmod N$ and that $\Pr[u = \pm v] = 1/2$. This implies that $\Pr[gcd(u - v, N) \in \{P, Q\}] = 1/2$ which enables $\mathcal{A}'$ to factor N. The complete proof follows the same lines along with additional "randomization" of the a_i's (achieved by taking $a_i = 2^{-1} + r_i$) which eliminates the assumption that $\mathcal{A}$ always succeeds.

Proof: Assume that $\mathcal{A}$ is as in Theorem 3.1, we define the algorithm $\mathcal{A}'$ that is guaranteed to exist by the theorem. Let $N = P \cdot Q$ be any $2n$-bit Blum-integer. Given N as its input, $\mathcal{A}'$ performs the following basic steps (we later describe how these steps can be carried out in the required running time):

1. Sample v uniformly at random in $\mathbb{Z}_N^*$. Compute $k = k(n)$ and $g = v^{2^k} \bmod N$. Denote by ℓ the order of g in $\mathbb{Z}_N^*$ (note that ℓ is not known to $\mathcal{A}'$).

   Since N is a Blum-integer and g is a quadratic-residue we have that ℓ is odd. This implies that $2 \in \mathbb{Z}_\ell^*$ and therefore $2^{-1} \bmod \ell$ exists.

2. Sample each one of the values in $\langle r_1, r_2, \ldots, r_k \rangle$ uniformly at random in $[N]$. For $1 \leq i \leq k$, denote by a_i the value $r_i + 2^{-1} \bmod \ell$ (again, note that a_i is not known to $\mathcal{A}'$). Denote by $\vec{a}$ the sequence $\langle a_1, a_2, \ldots, a_k \rangle$.

3. Invoke $\mathcal{A}$ with input $\langle N, g \rangle$ and answer each query, q, of $\mathcal{A}$ with the value $h^r_{N,g,\vec{a}}(q)$.

4. Given that $\mathcal{A}$ outputs the correct value — $h_{N,g,\vec{a}}(1^n)$, compute $u = g^{2^{-k}} \bmod N$. As will be noted below, $u^2 = v^2 \bmod N$. If $u \neq \pm v \bmod N$, output $gcd(u - v, N)$ which is indeed in $\{P, Q\}$. Otherwise, output 'failed'.

The Running-Time of $\mathcal{A}'$:
Steps (1) and (2) can easily be carried out in time $poly(n, k(n))$. For steps (3) and (4) to be carried out in time $t'(n) = poly(n, k(n)) \cdot t(n)$ it is enough to have that:

a. For every query $q \in \{0,1\}^k \setminus \{1^k\}$ the value $h_{N,g,\vec{a}}(q)$ can be computed in time $poly(n, k(n))$.

b. Given $h_{N,g,\vec{a}}(1^n)$, the value $g^{2^{-k}} \bmod N$ can be computed in time $poly(n, k(n))$.

The key-observation for showing the above is that for all $0 < i < k$, we have that $g^{2^{-i}} = v^{2^{k-i}} \bmod N$. For $i = 1$, this is implied by the fact that both $g^{2^{-1}}$ and $v^{2^{k-1}}$ are square roots of g and they are both quadratic-residues. Since squaring is a permutation over the set of quadratic-residues in $\mathbb{Z}_N^*$ (for any Blum-integer, N) we must have that $g^{2^{-1}}$ and $v^{2^{k-1}}$ are equal. By induction on $0 < i < k$, we get that $g^{2^{-i}} = v^{2^{k-i}} \bmod N$ in the same way. Therefore, for every $q = q_1 q_2 \ldots q_k \neq 1^k$:

$$h_{N,g,\vec{a}}(q) = g^{\prod_{q_i=1} a_i} = g^{\prod_{q_i=1}(r_i + 2^{-1})} = g^{\sum_{j=0}^{k-1} \alpha_j 2^{-j}} = v^{\sum_{j=0}^{k-1} \alpha_j 2^{k-j}} \bmod N$$

where the values $\{\alpha_j\}_{j=0}^{k-1}$ can easily be computed in time $poly(n, k(n))$. Therefore, we get that (a) holds. Similarly:

$$h_{N,g,\vec{a}}(1^k) = g^{\prod_{i=1}^{k} a_i} = g^{\prod_{i=1}^{k}(r_i+2^{-1})} = g^{2^{-k}} \cdot g^{\sum_{j=0}^{k-1}\beta_j 2^{-j}} = g^{2^{-k}} \cdot v^{\sum_{j=0}^{k-1}\beta_j 2^{k-j}} \bmod N$$

where the values $\{\beta_j\}_{j=0}^{k-1}$ can easily be computed in time $poly(n, k(n))$. We now get that (b) holds since:

$$g^{2^{-k}} = h_{N,g,\vec{a}}(1^k) \cdot \left(v^{\sum_{j=0}^{k-1}\beta_j 2^{k-j}}\right)^{-1} \bmod N$$

The Success-Probability of $\mathcal{A}'$:
It remains to show that $\mathcal{A}'$ ϵ'-factors, where $\epsilon'(n) = \epsilon(n)/2 - O(k(n) \cdot 2^{-n})$. Recall that u denotes the value $g^{2^{-k}} \bmod N$. As shown above $v^2 = g^{2^{-(k-1)}} (= u^2) \bmod N$. Therefore, it is not hard to verify that:

$$\Pr[\mathcal{A}'(N) \in \{P, Q\}] = \Pr\left[(u \neq \pm v \bmod N) \ \ and \ \ \left(\mathcal{A}^{h^r_{N,g,\vec{a}}}(N, g) = h_{N,g,\vec{a}}(1^n)\right)\right]$$

Note that $\mathcal{A}^{h^r_{N,g,\vec{a}}}(N, g)$ does not depend on v itself but rather on v^2 (i.e., $\mathcal{A}^{h^r_{N,g,\vec{a}}}(N, g)$ is equally distributed for any two assignments, u and v, of v as long as $u^2 = v^2 \bmod N$). We therefore get that:

$$\Pr[\mathcal{A}'(N) \in \{P, Q\}] = 1/2 \cdot \Pr[\mathcal{A}^{h^r_{N,g,\vec{a}}}(N, g) = h_{N,g,\vec{a}}(1^n)]$$

Let N be chosen from $FIG(1^n)$. We need to show that for infinitely many n's

$$\Pr[\mathcal{A}'(N) \in \{P, Q\}] > \epsilon'(n)$$

Which is equivalent to showing that for infinitely many n's

$$\Pr[\mathcal{A}^{h^r_{N,g,\vec{a}}}(N, g) = h_{N,g,\vec{a}}(1^n)] > \epsilon(n) - O(k(n) \cdot 2^{-n})$$

To do so, let us first review a couple of simple facts on the distribution of g and of each $a_i \bmod \ell$:

Fact1 g is a uniformly distributed quadratic-residue in $\mathbb{Z}_N^*$.

> **Reason:** v is a uniformly distributed quadratic-residue in $\mathbb{Z}_N^*$ and squaring is a permutation over the set of quadratic-residues in $\mathbb{Z}_N^*$ (since N is a Blum-integer).

Fact2 Let r and a' be uniformly distributed in $[N]$ and denote by a the value $r + 2^{-1} \bmod \ell$. Then a and $a' \bmod \ell$ are of statistical distance $O(2^{-n})$.

> **Reason:** ℓ divides $(Q-1)(P-1)$. Therefore the distribution of a conditioned on the event that $r \in [(Q-1)(P-1)]$ is the same as the distribution of $a' \bmod \ell$ conditioned on the event that $a' \in [(Q-1)(P-1)]$ (and in both cases it is simply the uniform distribution over $\mathbb{Z}_\ell$). It remains to notice that $\Pr[r \in [(Q-1)(P-1)]] = \Pr[a' \in [(Q-1)(P-1)]] = \frac{(Q-1)(P-1)}{N} = O(2^{-n})$.

Let each value in $\vec{a'} = \langle a'_1, a'_2, \ldots, a'_k \rangle$ be uniformly distributed in $[N]$. Since $\mathcal{A}$ ϵ-solves the GDH$_k$-Problem and given Fact1, we have that:

$$\Pr[\mathcal{A}^{h^r_{N,g,\vec{a'}}}(N,g) = h_{N,g,\vec{a'}}(1^n)] > \epsilon(n)$$

Given Fact2, it is easy to verify that the two random variables $h_{N,g,\vec{a}}$ and $h_{N,g,\vec{a'}}$ are of statistical distance $O(k(n) \cdot 2^{-n})$. Therefore, we can conclude that:

$$\Pr[\mathcal{A}^{h^r_{N,g,\vec{a}}}(N,g) = h_{N,g,\vec{a}}(1^n)] > \epsilon(n) - O(k(n) \cdot 2^{-n})$$

which completes the proof of the theorem. $\Box$

4 Conclusions

In this note it was proven that breaking the generalized Diffie-Hellman assumption modulo a Blum-integer is at least as hard as factoring Blum-integers. This implies that the security of the generalized Diffie-Hellman key-exchange protocol (which is mentioned in the introduction) can be reduced to the assumption that factoring is hard. In addition, as shown in [3], it implies the existence of efficient pseudo-random functions which are at least as secure as factoring.

A possible line for further research is the study of the generalized Diffie-Hellman assumption in other groups and the relation between the generalized Diffie-Hellman assumption and the standard Diffie-Hellman assumption. It is interesting to note that the decisional versions of the generalized Diffie-Hellman assumption and the standard Diffie-Hellman assumption are equivalent (as shown in [6]).

References

[1] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, vol. 22(6), 1976, pp. 644-654.

[2] K. McCurley, A key distribution system equivalent to factoring, *J. of Cryptology*, vol 1, 1988, pp. 95-105.

[3] M. Naor and O. Reingold, Number-Theoretic constructions of efficient pseudo-random functions, to appear in: *Proc. 38th IEEE Symp. on Foundations of Computer Science*, 1997.

[4] M. O. Rabin, Digitalized signatures and public-key functions as intractable as factorization, Technical Report, TR-212, MIT Laboratory for Computer Science, 1979.

[5] Z. Shmuely, Composite Diffie-Hellman public-key generating systems are hard to break, Technical Report No. 356, Computer Science Department, Technion, Israel, 1985.

[6] M. Steiner, G. Tsudik and M. Waidner, Diffie-Hellman key distribution extended to group communication, *Proceedings 3rd ACM Conference on Computer and Communications Security*, 1996, pp. 31-37.