# On a Key Exchange Protocol

Mugurel Barcau[1,2], Vicenţiu Paşol[1,2], Cezar Pleşca[1,3], and Mihai Togan[1,3(✉)]

[1] certSIGN - Research and Development, Bucharest, Romania
barcau@yahoo.com, vpasol@yahoo.com, cezar.plesca@gmail.com, mihai.togan@gmail.com
[2] Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Bucharest, Romania
[3] Military Technical Academy, Bucharest, Romania

**Abstract.** In this paper we investigate an instance of the generalized Diffie-Hellman key exchange protocol suggested by the equidistribution theorem. We prove its correctness and discuss the security. Experimental evidences for the theoretical results are also provided.

**Keywords:** Key exchange protocol · Diffie-Hellman · Rational approximations

## 1 Introduction

The question of key exchange is a fundamental problem in the areas of cryptography and communication security. The key exchange protocols are cryptographic primitives used to set up shared secret keys in order to enable secure communication over unreliable networks. They are the most used cryptographic tools in building secure communication protocols over the Internet (e.g. IPsec, SSH, and TLS). The first practical method for establishing such a shared secret was the Diffie-Hellman key agreement protocol, which was introduced in [3]. Much later, a generalized Diffie-Hellman algorithm was defined as a general tool for generating key exchange protocols (see [7]). The idea is very simple in essence and it can be stated as follows: assume there exist a commutative semigroup $G$ and a set $X$ such that $G$ acts on $X$, and the action of $G$ cannot be inverted; in the sense that if one has $x \in X$, and $g \cdot x$ (where $g \in G$), finding $g$ is a hard task (cannot be done in polynomial time). Then, the Diffie-Hellman algorithm runs as follows: an element $x \in X$ is publicly given. Alice and Bob each choose at random secret private keys $a \in G$, $b \in G$, respectively. Alice sends $a \cdot x$ to Bob and Bob sends $b \cdot x$ to Alice. Then, Alice computes $a \cdot (b \cdot x)$, while Bob computes $b \cdot (a \cdot x)$. The associativity of the action of $G$ on $X$, and the commutativity of $G$ imply that both Alice and Bob will arrive to the same result which is set to be the common key. In this article, we investigate the key exchange protocol resulted from the action of the monoid $\mathbb{N}$ of the set of natural numbers with multiplication on the closed-open interval $[0, 1)$ of the set of real numbers, given by $(c, x) \mapsto \{cx\}$, where $\{y\}$ is the fractional part of $y \in \mathbb{R}$. Since real numbers

© Springer International Publishing AG 2017
P. Farshim and E. Simion (Eds.): SecITC 2017, LNCS 10543, pp. 187–199, 2017.
https://doi.org/10.1007/978-3-319-69284-5_13

with infinite binary representations cannot be practically used in computational algorithms, one has to use approximations of their fractional parts, so that one discovers that an approximation of this action may be the truncated product $(c, x) \mapsto \left\lfloor \frac{c \cdot x \pmod{2^n}}{2^m} \right\rfloor$ (here, $y \pmod{2^n}$ is the remainder after division of $y$ by $2^n$). A different property of this function (with $x$ constant), more precisely its "approximately" linearity, has been used by R. Merkle in [8] to construct a key exchange protocol, which is distinct from ours.

On the other hand, a variant of the protocol constructed in this article appears in [1]. However, the authors of [1] show only experimentally that Alice and Bob get the same common key with great probability; our protocol is proven to work in all cases. We also show that the protocol is a particular instance of the generalized Diffie-Hellman key exchange protocol. Moreover, the security reduction in [1] seems faulty to us and we give two ways of showing that in fact the security of the protocol is much stronger than the security suggested in [1] (we support our conclusions also by experimental results); in particular the security reductions we construct are strong arguments for believing that the protocol is in fact a quantum-resistant protocol, which is yet another advantage over the classical Diffie-Hellman protocols, over the sought advantage of being much more efficient with respect to computational complexity.

The article is structured as follows: in Sect. 1 we present the general construction of the Diffie-Hellman protocol, as it appears in [7]. In the next section we discuss the $\mathbb{N}$-monoid action described above. We shall also explain how starting with this monoid action one ends up with the truncated product function. Section 3 contains the proof of our main result, and as a consequence we describe the resulted key exchange protocol. In Sect. 4 we discuss the security of our protocol giving the necessary sizes of the parameters for practical implementation. In the next section we give experimental evidence of the fact that the truncated product function used in the algorithm cannot be inverted and also some applications of the protocol. We end the paper with a section that contains several conclusions.

## 2 The Generalized Diffie-Hellman Protocol

Let $G$ be a semigroup, that is a set with an associative binary operation, denoted by "$\cdot$". In particular, we do not require that $G$ has an identity element. The semigroup $G$ is abelian if the operation is commutative. If $S$ is a set, *an action of $G$ on $S$* is a map

$$\phi : G \times S \longrightarrow S$$

such that $\phi(g \cdot h, s) = \phi(g, \phi(h, s)), \forall g, h \in G, s \in S$. If $G$ is a monoid, i.e. it has identity element 1, then we shall require that $\phi(1, s) = s, \forall s \in S$. In general, we shall denote $\phi(g, s)$ by $g \cdot s$, and refer to such an action as a $G$-action on the set $S$ and to $S$ as a $G$-set.

We now present the key exchange protocols based on semigroup actions, as they where introduced in [7].

**Protocol 1** *(Diffie-Hellman Key Exchange Protocol). Let $S$ be a finite set, and let $G$ be an abelian semigroup acting on $S$. The Diffie-Hellman Key Exchange Protocol based on the $G$-set $S$ is the following protocol:*

1. *Alice and Bob publicly agree on an element $s \in S$.*
2. *Alice chooses $a \in G$ and computes $a \cdot s$. Alice's private key is $a$, and her public key is $a \cdot s$.*
3. *Bob chooses $b \in G$ and computes $b \cdot s$. Bob's private key is $b$, and his public key is $b \cdot s$.*
4. *Their common secret key is*

$$a \cdot (b \cdot s) = (a \cdot b) \cdot s = (b \cdot a) \cdot s = b \cdot (a \cdot s)$$

The above protocol is secure only if the following problem is hard:

*Problem 1* (Semigroup Action Problem). Given a semigroup $G$ acting on a set $S$, and elements $x \in S$ and $y \in Gx$, find $g \in G$ such that $g \cdot x = y$.

If an attacker, Eve, can find $a' \in S$ such that $a' \cdot s = a \cdot s$, then she finds the shared secret by computing: $a' \cdot (b \cdot s) = b \cdot (a' \cdot s) = b \cdot (a \cdot s)$.

*Problem 2* (The Diffie-Hellman Semigroup Action Problem). Given an abelian semigroup $G$ acting on a finite set $S$, and elements $x, y, z \in S$ with $y = g \cdot x$ and $z = h \cdot x$ for some $g, h \in G$, find $(g \cdot h) \cdot x \in S$.

It is clear that the security of the above protocol is equivalent to this problem. On the other hand, the only way known to attack the Diffie-Hellman Semigroup Action Problem is by solving the Semigroup Action Problem. It is unknown if these two problems are equivalent. We refer to *loc.cit.* for a detailed discussion about the generic attacks on the Semigroup Action Problem.

## 3 Case Study

### 3.1 Irrational Numbers and Equidistribution Theorem

The idea behind the Diffie-Hellman algorithm that we will study in this article is based on the well known equidistribution theorem which asserts that if $x$ is an irrational number, then the set $\{\{n \cdot x\} \mid n \in \mathbb{N}\}$ is uniformly distributed in the interval $(0, 1)$, where $\{x\}$ stands for the fractional part of the real number $x$ (see, [2,13,15].) Moreover, the monoid of natural numbers with multiplication acts on the interval $[0, 1)$ via the formula suggested by the equidistribution theorem $\mathbb{N} \times [0, 1) \mapsto [0, 1)$, $(n, x) \mapsto \{n \cdot x\}$:

It is very easy to check that indeed,

$$\{m \cdot \{n \cdot x\}\} = \{mn \cdot x\} = \{n \cdot \{m \cdot x\}\}.$$

There are two issues to be resolved concerning this example. The first one is how do we represent the irrational numbers in order to do practical computation.

And the second issue (which is obviously related to the first one) is how certain we are that the corresponding Diffie-Hellman algorithm is secure. There are two alternatives to represent a real number: by symbols or by approximation.
The first alternative implies that the number is considered as a solution of certain algebraic/differential equations, e.g. we represent $\sqrt{2}$ as the unique positive solution of the equation $x^2-2=0$. This type of representation is not suitable for our purposes since the representation of $\{n\cdot x\}$ would reveal $n$, or, even worse, would be impossible even to compute $\{n\cdot x\}$ for large $n$.
On the other hand the alternative of approximating a real number (by some finite expression) seems to be also doomed since that representation would actually represent a rational number (in any of the natural known representations, i.e. by digits in some base, by continued fraction, etc.). But then, we will loose the nice property of uniform distribution of the numbers $\{n\cdot x\}$ when $n$ varies, which should be important for proving the randomness of the algorithm.

We will choose the second alternative and see that in fact, the randomness property is not entirely lost, but rather propagates well enough to prove the security of the algorithm.

### 3.2 Base 2 Approximation of Subunitary Numbers and Merkle's Approximately Linear Hash Function

We choose base 2 approximation because it is the most suited for computer manipulations. Let $n$ be a natural number (to be setup later) and consider an irrational number $x\in(0,1)$. We write $\bar{x}=\bar{x}_n\in\{0,\dots,2^n-1\}$ its base 2, $n$-digit expansion i.e. $\bar{x}=\lfloor 2^n x\rfloor$, where $\lfloor y\rfloor$ stands for the integer part of a real number $y$. In general, for any positive real number, we write $\bar{x}=\bar{x}_n:=\lfloor 2^n x\rfloor \pmod{2^n}$. We omit the index $n$ in the notation of $\bar{x}$ if it is obvious from the context. If $a$ is a $k$-digit number ($k\le n$), then the $(n-k)$-bit expansion of $\{ax\}$ will be almost $\lfloor(a\bar{x}\pmod{2^n})/2^k\rfloor$, where by $a\bar{x}\pmod{2^n}$ we mean the remainder from division of $a\bar{x}$ by $2^n$. One should notice that the function $a\mapsto\lfloor(a\bar{x}\pmod{2^n})/2^k\rfloor$ is the approximately linear function $AL(a,x)$ in [8]. It will become clear in the next sections how good is the last approximation (it can differ by 1 at most). Observe that if we publicly publish $x$ of size $n$, then $\overline{\{ax\}}_n$ determines $a$ if small enough, thus we cannot use this function for a key exchange protocol. However, if we cut the last $k$ digits, where $k$ is the size of $a$, the function becomes not invertible, as we shall see in the section dedicated to the security of the protocol. We have now in our hands, indeed the tools (commutative semigroup action and hardness of semigroup action problem) in order to produce the Diffie-Hellman protocol.

## 4 Key Exchange Protocol

For any $a\in[0,2^k-1]$, $x\in[0,2^n-1]$ and positive integer $m\le n$ we define the function

$$\phi_{(k,n,m)}(a,x):=\left\lfloor 2^m\left\{\frac{ax}{2^n}\right\}\right\rfloor=\left\lfloor\frac{ax\pmod{2^n}}{2^{n-m}}\right\rfloor$$

(here, as before, by $ax \pmod{2^n}$ we mean the remainder from division of $ax$ by $2^n$).

**Theorem 2.** *Let $k, n, m, l$ be positive integers such that $m \geq k + l$. For any $a, b \in [0, 2^k - 1]$, $x \in [0, 2^n - 1]$ there exists $\delta \in \{-1, 0, 1\}$ such that*

$$\phi_{(k,m,l)}(a, \phi_{(k,n,m)}(b, x)) \equiv \phi_{(k,m,l)}(b, \phi_{(k,n,m)}(a, x)) + \delta \pmod{2^l}$$

*Proof.* We make the following notations:

$$x_A := \phi_{(k,n,m)}(a, x) \in [0, 2^m - 1], x_B := \phi_{(k,n,m)}(b, x) \in [0, 2^m - 1]$$

Since $\dfrac{ax}{2^n} = \left\lfloor \dfrac{ax}{2^n} \right\rfloor + \left\{ \dfrac{ax}{2^n} \right\}$ we get:

$$\frac{ax}{2^{n-m}} = 2^m \left\lfloor \frac{ax}{2^n} \right\rfloor + 2^m \left\{ \frac{ax}{2^n} \right\} = 2^m \left\lfloor \frac{ax}{2^n} \right\rfloor + x_A + \left\{ 2^m \left\{ \frac{ax}{2^n} \right\} \right\},$$

which yields the inequalities:

$$0 \leq \frac{ax}{2^n} - \left\lfloor \frac{ax}{2^n} \right\rfloor - \frac{x_A}{2^m} < \frac{1}{2^m}$$

Now, denote by $x_{AB} := \phi_{(k,m,l)}(a, x_B)$, and by $x_{BA} := \phi_{(k,m,l)}(b, x_A)$, then we have:

$$\begin{aligned}
&0 \leq \frac{abx}{2^n} - b\left\lfloor \frac{ax}{2^n} \right\rfloor - \frac{bx_A}{2^m} < \frac{b}{2^m} \\
&0 \leq \frac{abx}{2^n} - b\left\lfloor \frac{ax}{2^n} \right\rfloor - \left\lfloor \frac{bx_A}{2^m} \right\rfloor - \left\{ \frac{bx_A}{2^m} \right\} < \frac{b}{2^m} \\
&0 \leq \frac{abx}{2^{n-l}} - 2^l \left( b\left\lfloor \frac{ax}{2^n} \right\rfloor + \left\lfloor \frac{bx_A}{2^m} \right\rfloor \right) - 2^l \left\{ \frac{bx_A}{2^m} \right\} < \frac{b}{2^{m-l}} \\
&0 \leq \frac{abx}{2^{n-l}} - 2^l \left( b\left\lfloor \frac{ax}{2^n} \right\rfloor + \left\lfloor \frac{bx_A}{2^m} \right\rfloor \right) - \left\lfloor 2^l \left\{ \frac{bx_A}{2^m} \right\} \right\rfloor - \left\{ 2^l \left\{ \frac{bx_A}{2^m} \right\} \right\} < \frac{b}{2^{m-l}} \\
&0 \leq \frac{abx}{2^{n-l}} - 2^l \left( b\left\lfloor \frac{ax}{2^n} \right\rfloor + \left\lfloor \frac{bx_A}{2^m} \right\rfloor \right) - x_{BA} < \frac{b}{2^{m-l}} + \left\{ 2^l \left\{ \frac{bx_A}{2^m} \right\} \right\}
\end{aligned}$$

Since $m \geq k + l$, we deduce $0 \leq \dfrac{b}{2^{m-l}} + \left\{ 2^l \left\{ \dfrac{bx_A}{2^m} \right\} \right\} < 2$, so that:

$$\left\lfloor \frac{abx}{2^{n-l}} \right\rfloor = 2^l \alpha_{BA} + x_{BA} + \epsilon_{BA} \tag{1}$$

for some integer $\alpha_{BA}$ and $\epsilon_{BA} \in \{0, 1\}$. Similarly we have:

$$\left\lfloor \frac{abx}{2^{n-l}} \right\rfloor = 2^l \alpha_{AB} + x_{AB} + \epsilon_{AB} \tag{2}$$

for some integer $\alpha_{AB}$ and $\epsilon_{AB} \in \{0, 1\}$. From (1) and (2) we get the congruence:

$$x_{AB} \equiv x_{BA} + \delta \pmod{2^l}$$

where $\delta \in \{-1, 0, 1\}$.

Notice that if $l \geq 2$ the last congruence gives $x_{AB} \equiv x_{BA} + \delta \pmod{2^2}$, which means that the last two digits in the binary decompositions of $x_{AB}$ and $x_{BA}$ determine $\delta$. This simple but important observation is included in the following key exchange protocol.

1. **Public key:** Choose $n, k, m, l$ such that $n \geq m \geq k + l$. Pick a random *good* number $x \in [0, 2^n - 1]$. The public key is $(n, k, m, l, x)$.
2. **Secret choices:** Alice picks a random number $a \in [0, 2^k - 1]$ and Bob picks a random number $b \in [0, 2^k - 1]$.
3. **Exchange:** Alice computes $x_A := \left\lfloor \frac{ax \pmod{2^n}}{2^{n-m}} \right\rfloor$ and Bob computes $x_B := \left\lfloor \frac{bx \pmod{2^n}}{2^{n-m}} \right\rfloor$. Alice sends $x_A$ to Bob and Bob sends $x_B$ to Alice.
4. **Verify key:** Alice computes $x_{AB} := \left\lfloor \frac{ax_B \pmod{2^m}}{2^{m-l}} \right\rfloor$, then she sets $(k_A, v_A)$ to be the most significant $l-2$ digits, respectively the least significant 2 digits of $x_{AB}$. Similarly, Bob computes $(k_B, v_B)$. Both publicly publish $v_A$ and $v_B$, respectively.
5. **Common key:** If $(v_A, v_B) = (00, 11)$ then the common key is $K := k_A = (k_B + 1) \pmod{2^{l-2}}$. If $(v_A, v_B) = (11, 00)$ then the common key is $K := (k_A + 1) \pmod{2^{l-2}} = k_B$. Otherwise, the common key is $K := k_A = k_B$.

By a *good* number, we mean an odd number whose distribution of $0's$ and $1's$ in its binary expansion is random.

Notice that the value $n - m$ must be large enough to be resistant to brute force attacks, thus from the security perspective, the choice of $n$, $m$, and $l$ has to be such that $n \geq m + k \geq 2k + l$.

### 4.1 Security

As explained in Sect. 2, the security of our protocol is based on the hardness of inverting the function $a \mapsto \left\lfloor \frac{ax \pmod{2^n}}{2^{n-m}} \right\rfloor$, where $x$ is a known (good) $n$-digit number. The authors in [1] suggest that the hardness of this problem can be reduced to an instance of SAT by explicitly writing down the equations for the digits of $a$ and the carry-overs, and comparing those equations with the equations used in [5] to instantiate SAT from FACT which is believed to be classically hard. However, Merkle's 3SAT reduction, see [8], suggests that the hardness of the problem is in fact based on an NP complete problem, which indicates that the problem might be also quantum secure. Our personal take is towards Merkle's point of view. Moreover, the authors in [1] seem to overlook some facts about the shape of the equations in their comparison with FACT. To give a stronger argument why we are inclined towards Merkle's point of view, we argue that the problem is more related to CVP (closest vector problem in a lattice) than to FACT. Let $y = \left\lfloor \frac{ax \pmod{2^n}}{2^{n-m}} \right\rfloor$, then there exist $q \in \mathbb{Z}$ and

$r \in [0, 2^{n-m} - 1]$ such that

$$\frac{ax}{2^n} - q = \frac{y}{2^m} + \frac{r}{2^n}.$$

Thus, one has to find the closest vector to $\frac{y}{2^m}$ in the "lattice" $\tilde{\Lambda}_x := \mathbb{Z} \cdot \frac{x}{2^n} + \mathbb{Z}$. The fact that the point $\frac{y}{2^m} + \frac{r}{2^n}$ is (probabilistically) the closest vector in $\tilde{\Lambda}_x$ for sufficiently random public key $x$ is implied by the fact that the function $a \mapsto y$ is probabilistically injective as shown bellow.

The "lattice" $\tilde{\Lambda}_x$ is "an approximation" of the lattice $\Lambda_\alpha := \mathbb{Z}\alpha + \mathbb{Z}$, for $\alpha$ an irrational number. Notice that the later lattice corresponds to a noncommutative elliptic curve $E_\alpha := \mathbb{R}/\Lambda_\alpha$ (see [14]) and the action of the monoid of natural numbers acts as usual multiplication (successive additions) on $E_\alpha$. One may argue now that in CVP the dimension of the lattice is important for the hardness of the problem. Note that in fact, by approximating the lattice and taking only the most important bits in this approximation, the "dimension" of the "lattice" can be considered to be $2k$ (this is the number of free variables over $\mathbb{F}_2$), which is in agreement with the usual setup of CVP.

### 4.2 Hashing Perspective

Another perspective over the security of the exchange protocol could be seen by considering Alice's computation $x_A$ as a multiplicative hash function of key $a \in [0, 2^k - 1]$, where $x$, $n$ and $m$ are constant parameters:

$$x_A := \left\lfloor \frac{ax \pmod{2^n}}{2^{n-m}} \right\rfloor$$

A simplified version of the general multiplicative hash function was proposed by Dietzfelbinger et al. [9] and consists in obtaining a hash value of size $m$ for an integer key $a \in [0, 2^n - 1]$, using the previous formula: $h(a) = x_A$. The authors show that if $x$ is a randomly chosen odd integer in $[0, 2^n - 1]$, then the collision probability of two different keys is almost $2/2^m$, which is a factor of two larger than what one could expect with a random function from $2^n \to 2^m$.

Now, let's consider $a \in [0, 2^k - 1]$ as a fixed value. Given the above mentioned collision probability, it means that if we randomly choose $a' \in [0, 2^n - 1]$, the probability to collude with $a$ (i.e. $h(a) = h(a')$) is almost $2/2^m$. One can easily notice that, knowing the approximately linear behavior of the hash function, if we restrain $a'$ to the range $[0, 2^k - 1]$, the collision probability remains $2/2^m$.

Let $X$ be a random variable that counts the number of collisions for function $h$ in the range $[0, 2^k - 1]$. Using a technique similar to the analysis of the birthday paradox [10], the expected number of collisions is limited to:

$$E[X] \leq \binom{2^k}{2} \frac{2}{2^m}$$

Now applying Markov's inequality, we have: $\text{Prob}(X \geq 1) \leq E[X]$. This implies that one can approximate the probability of no collision in all $2^k$ keys:

$$\text{Prob(no collision)} = \text{Prob}(X = 0) = 1 - \text{Prob}(X \geq 1) \geq 1 - E[X]$$

$$\implies \text{Prob(no collision)} \geq 1 - \binom{2^k}{2}\frac{2}{2^m} = 1 - \frac{2^k(2^k-1)}{2}\cdot\frac{2}{2^m} > 1 - 2^{m-2k}$$

Therefore, taking for instance $m \geq 3k$, the hash function $h$ becomes probabilistically injective, which implies that no further reductions can be made by an attacker on a brute force verification over the range $[0, 2^k - 1]$. Finally, we can conclude that the security parameter of the presented protocol is $k$.

However, the experiments show that in fact the conclusions of this subsection are in fact valid for smaller values, i.e. $m = 2k + 2$(see the bellow experimental results).

## 5 Experiments

### 5.1 Key Distribution

An important issue in the application of this protocol in practice is related to the randomness of the common secret key. In order to test this property, we conduct an experiment using a library for arbitrary-precision integer math, namely BIGNUM library that comes with OpenSSL [11].

As we want to obtain shared secret keys of length 128 bits, we have chosen the following values for the scheme parameters: $k = 128$, $l = k + 2 = 130$, $m = 2k + 2 = 258$ and $n = 3k + 2 = 386$. We pick three random numbers (using three independent generators): $x \in [0, 2^n - 1]$, $a \in [0, 2^k - 1]$ (Alice choice) and $b \in [0, 2^k - 1]$ (Bob choice). Then, using the protocol described in Sect. 4, we compute the shared secret key $s = k_A = k_B$. For such an execution, we also count if the protocol needs an additional step at the end to adjust the keys of Alice or Bob. We called this a key adjustment.

As the number of possible common keys is very large, we divide the range $[0, 2^{128} - 1]$ into 128 equal bins. We repeat the previous execution a number of $N = 10^8$ times, and for each execution, we place the secret key into its corresponding bin. The percentage of key adjustment cases is about $6,6\%$. Finally, we count the number of keys belonging to each bin, and normalize these frequencies. As expected, the keys distribution is almost uniform as shown in Fig. 1. The mean square error between our distribution and the ideal one is approximatively $7 * 10^{-9}$.

In order to verify the theoretical results presented in Sect. 4.2, we conduct another experiment, to obtain the collision probability for the hash function $h(a) = x_A : [0, 2^k - 1] \to [0, 2^m - 1]$. We vary the length of the shared key $k$ in the range $[10 - 20]$; the other parameters are computed as previously: $l = k + 2$, $m = 2k + 2$ and $n = 3k + 2$.

For each $k$, using three independent random generators for $x \in [0, 2^n - 1]$, $a \in [0, 2^k - 1]$ and $b \in [0, 2^k - 1]$, we compute $x_A$ and $x_B$. Theoretically, we'll have a collision (i.e. $x_A = x_B$) with a probability inferior to $1/2^{m-1}$. We execute this experience a number of $N = 2^{m-1} \cdot 1000$ times, count the number of collisions and convert this number into a probability.

The results obtained are illustrated in Fig. 2, using a logarithmic scale for collision probability. One can notice that the collision probability is a factor of

two smaller than the theoretical limit $1/2^{m-1}$. In other words, this empirical probability (i.e. $1/2^m$) is equivalent to what one could expect with a random function from $2^k \to 2^m$. Building on this result, we can assume that the function $x_A$ is statistically injective, which confirms the theoretical results from Sect. 4.2.

### 5.2 Rough Distributions

We will show in what follows why the probability of key adjustment agrees with the empirical data we produced.

As in the previous sections, we have the following ($m = n - k$):

$$0 \leq \frac{abx}{2^{n-l}} - 2^l \left( b \left\lfloor \frac{ax}{2^n} \right\rfloor + \left\lfloor \frac{b \cdot x_A}{2^{n-k}} \right\rfloor \right) - x_{BA} = \frac{b \cdot r_A}{2^{n-l}} + \left\{ 2^l \left\{ \frac{b \cdot x_A}{2^{n-k}} \right\} \right\}.$$

Taking the integer parts, we get:

$$\left\lfloor \frac{abx}{2^{n-l}} \right\rfloor - 2^l \left( b \left\lfloor \frac{ax}{2^n} \right\rfloor + \left\lfloor \frac{b \cdot x_A}{2^{n-k}} \right\rfloor \right) = x_{BA} + \frac{b \cdot r_A}{2^{n-l}} + \left\{ 2^l \left\{ \frac{b \cdot x_A}{2^{n-k}} \right\} \right\}.$$

Modulo $2^l$ the left hand side is the same for Alice and Bob, therefore for a key adjustment to take place, the right hand side has to change. Since

$$\left\{ 2^l \left\{ \frac{b \cdot x_A}{2^{n-k}} \right\} \right\} = \left\{ \frac{abx \pmod{2^n}}{2^{n-l}} - \frac{b \cdot r_A}{2^{n-l}} \right\}.$$

Thus in order to have a change on the right hand side either $\left\{ \frac{abx \,(\mathrm{mod}\ 2^n)}{2^{n-l}} - \frac{b \cdot r_A}{2^{n-l}} \right\} + \frac{b \cdot r_A}{2^{n-l}} > 1$ and $\left\{ \frac{abx \,(\mathrm{mod}\ 2^n)}{2^{n-l}} - \frac{a \cdot r_B}{2^{n-l}} \right\} + \frac{a \cdot r_B}{2^{n-l}} < 1$ or the other way around.

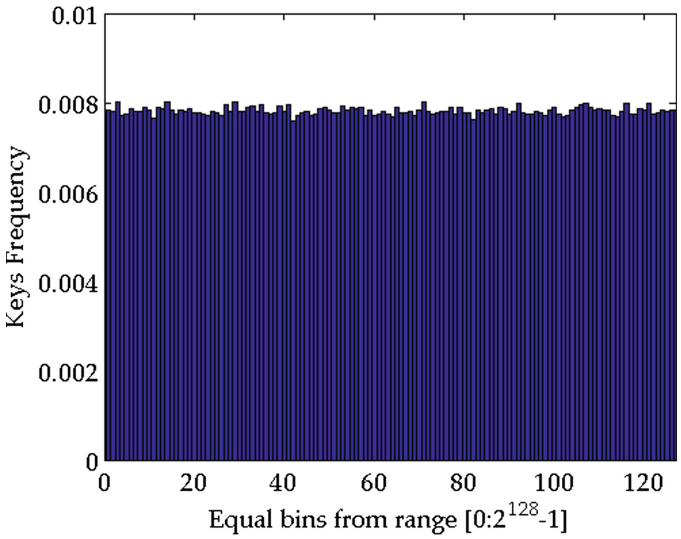


**Fig. 1.** Shared secret keys distribution

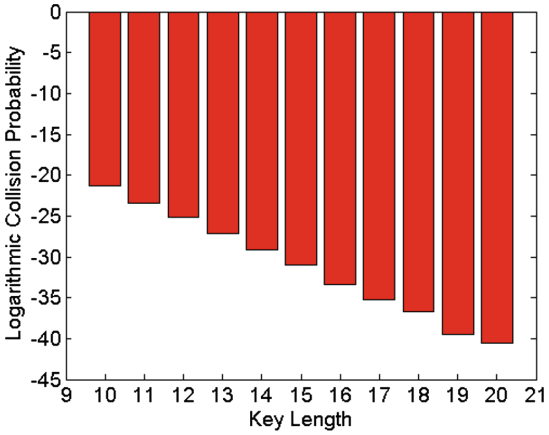


**Fig. 2.** Collision probability of hash function

Notice that $\{u-v\}+v>1 \Leftrightarrow \{u\}<v$ for any real $u$ and $0<v<1$. We get:

$$\frac{a \cdot r_B}{2^{n-l}} \leq \left\{ \frac{abx \pmod{2^n}}{2^{n-l}} \right\} < \frac{b \cdot r_A}{2^{n-l}},$$

or the other way around. That means that $abx \pmod{2^n}$ sits in a short interval. As one can see, the length of this interval depends upon the choice of $a$ and $b$. However, on average (over $x$ good), the length of the interval is around $\frac{|b-a|}{2^{n-l}}$. Using the uniform distribution of $abx \pmod{2^n}$ when $x$ varies, we get that $P_{a,b}(change) \simeq \frac{|b-a|}{2^k}$, where by $P_{a,b}(change)$ we mean the probability of having different $x_{AB}$ and $x_{BA}$ fixed secret keys (only the public key is variable). We should be careful not to double count since $P_{a,b}(change) = P_{b,a}(change)$ (the interval is symmetric in the secret keys thus the same $x$ is valid for change when we swipe the secret keys.). The total average would be:

$$P(change) = \frac{1}{2^{2k}} \frac{1}{2} \sum_{a,b} P_{a,b}(change) \simeq \frac{1}{2^{3k+1}} \sum_{a,b} |b-a| = \frac{1}{2^{3k}} \sum_{c} \frac{c(c+1)}{2},$$

The last sum equals $\frac{1}{2^{3k}} \frac{M(M+1)(M+2)}{3}$ where $M = 2^k - 1$ is the range of summation. This gives a rough approximation of $P(change) = \frac{1}{3}$.

Finally, notice that the key adjustment occurs only $1/4$ of the times since out of the 8 possibilities for $(v_{AB}, v_{BA})$, only 2 of them produce key adjustment. Thus the rough probability (on average) of key adjustment would be around $1/12$. We have to be notice that in practice, this probability is in fact smaller since the choice of the public key is not in fact random, but it has to be a random number so it comes from a good approximation of an irrational number.

It seems a difficult task to compute the exact probability of key adjustment and we leave this computation as an open question.

### 5.3 Comparison with DLP-Based Diffie-Hellman Protocol

An important issue is also the computation effort required to run the protocol phases. This could have severe implications, especially in the case of resources with low power consumption requirements (like IoT devices) or in the cases where the millions of secret keys must be exchanged during a short time period. The key exchange protocol presented above is much less computing intensive than Diffie-Hellman protocol based on classical discrete logarithm problem over $Z_p$. To achieve a common secret, each party has to make only two multiply operation on integers, the first one to generate the information $X_A$ that has to be exchanged, and the second one to compute the verify key $X_{AB}$. Truncations are also used to discard the first and the last k-bits of the result, but these are almost free in terms of computing costs. Besides this, the initial setup of the protocol required to generate the parameters and the public/secret keys are also free in terms of computational effort.

For comparison purpose, we conducted few experiments intended to estimate the computation efforts required by our key-exchange protocol by comparison with DLP-based DH and ECDH protocols. We used an OpenSSL BIGNUM library based implementation for our protocol and the reference implementations for DH and ECDH variants included also in the OpenSSL package.

To have a relevant comparison, we have used similar configurations for the security-bits level (it is established by $k$ value in our protocol), and identical length for the computed secret keys. For an equivalent security parameter $k$ and a given length $L_{SK}$ for the outputted secret key, we instantiated the protocol with a setup based on the following parameters: $l = L_{SK} + 2$, $m = k + L_{SK} + 2$ and $n = 2k + L_{SK} + 2$. To estimate the required computation effort, we measured the costs in terms of the processing time of the steps that compute the common secret key. The initial phases of the protocol required for the generation of the parameters, public and private keys have not been taken into account.

In the case of DH protocol over $\mathbb{F}_p$ and using a modulus $p$ of 2048-bits length (this leads to a security level of $k = 112$-bits [12]), our protocol was 4500 times faster. In the case of DH with a modulus of 3072-bits length (security level of 128-bits), our protocol was 10000 times faster. Even DH on elliptic curves is less expansive that DH on $\mathbb{F}_p$, our key-exchange variant is about 1000 times faster than ECDH on prime fields $\mathbb{F}_p$, and respectively about 2000 times faster than ECDH on binary fields $\mathbb{F}_{2^m}$. Our experiments were conducted on a machine based on Intel(R) Xeon(R) E5-1620 at 3.60GHz CPU. In the case of much slower computing resources (such as IoT enabling devices), we expect that the mentioned speed-up rates will be much higher.

## 6 Conclusions

The observations in Sect. 4.1 might suggest new types of key exchange protocols by considering rational approximations for other geometric meaningful encryption algorithms. The advantage of rational approximations would be two folded: computational complexity relaxation of the usual encryption algorithms

and additional security via a CVP-type argument. The theoretical and experimental results in this paper make our protocol valuable for the cases where slow computing resources are available as mentioned above. Further work will concentrate on improving the theoretical security bounds as suggested by the experimental results and on the practical implementation of the protocol.

After the write-up of the paper we learned that there exists an attack of the key exchange protocol presented in this paper using an embedding of the security problem of our protocol into a two dimensional Closest Vector Problem (see for instance [16]). More precisely, one has a precise description of a two dimensional lattice and the constraints in our protocol ask for finding a lattice vector of bounded distance. Moreover, the exact conditions imposed on the parameters imply that one has a unique solution, thus one can test the enumerated lattice vectors output by the Bounded Distance Decoding algorithm (see [6]) and find a solution of our security problem. Unfortunately, one cannot modify the parameters in our protocol so the attack becomes unfeasible.

**Acknowledgments.** This research was partially supported by the Romanian National Authority for Scientific Research (CNCS-UEFISCDI) under the project PN-III-P2-2.1-PTE-2016-0191.

## References

1. Azhari, A., Bouftass, S.: On a new fast public key cryptosystem. https://eprint.iacr.org/2014/946.pdf
2. Bohl, P.: Über ein in der Theorie der säkutaren Störungen vorkommendes Problem. J. Reine Angew. Math. **135**, 189–283 (1909)
3. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
4. Gerold Grünauer Proposal of a new efficient public key system for encryption and digital signatures. https://eprint.iacr.org/2007/445.pdf
5. Horie, S., Watanabe, O.: Hard instance generation for SAT. In: Leong, H.W., Imai, H., Jain, S. (eds.) ISAAC 1997. LNCS, vol. 1350, pp. 22–31. Springer, Heidelberg (1997). doi:10.1007/3-540-63890-3_4
6. Liu, Y.-K., Lyubashevsky, V., Micciancio, D.: On bounded distance decoding for general lattices. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX/RANDOM -2006. LNCS, vol. 4110, pp. 450–461. Springer, Heidelberg (2006). doi:10.1007/11830924_41
7. Maze, G., Monico, C., Rosenthal, J.: Public key cryptography based on semigroup actions. Adv. Math. Commun. **1**(4), 489–507 (2007)
8. Merkle, R.C.: Public key distribution using approximately linear functions. http://www.merkle.com/papers/approxLinearPK.html
9. Dietzfelbinger, M., Hagerup, T., Katajainen, J., Penttonen, M.: A reliable randomized algorithm for the closest-pair problem. J. Algorithms **25**(1), 19–51 (1997)
10. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
11. Serpette, B., Vuillemin, J., Hervé, J.-C.: BigNum: a portable and efficient package for arbitrary-precision arithmetic. Digital, Paris Research Laboratory (1989)

12. NIST: Recommendation for Key Management, NIST Special Publication 800–57 Part 1 Revision 4 2016
13. Sierpinski, W.: Sur la valeur asymptotique d'une certaine somme. Bull Intl. Acad. Polonmaise des Sci. et des Lettres (Cracovie) series A, 9–11 (1910)
14. Soibelman, Y.: Quantum tori mirror symmetry and deformation theory. Lett. Math. Phys. **56**(2), 99–125 (2001)
15. Weyl, H.: Über die Gibbs'sche Erscheinung und verwandte Konvergenzphänomene. Rendiconti del Circolo Matematico di Palermo, pp. 377–407 (1910)
16. Zhang, Y.: A practical attack to Bouftass's crypto system. https://arxiv.org/abs/1605.00987v1