

QRKE: Quantum-Resistant Public Key Exchange

G. Brands¹, C.B. Roellgen², K.U. Vogel³

09.29.2015

Abstract

A “Post-Quantum Key Exchange” is needed since the availability of quantum computers that allegedly allow breaking classical algorithms like Diffie-Hellman, El Gamal, RSA and others within a practical amount of time is broadly assumed in literature. Although our survey suggests that practical quantum computers appear to be by far less advanced as actually required to break state-of-the-art key negotiation algorithms, it is of high scientific interest to develop fundamentally immune key negotiation methods. A novel polymorphic algorithm based on permutable functions and defined over the field of real numbers is proposed. The proposed key exchange can operate with at least four different strategies. The cryptosystem itself is highly variable and, due to the fact that rounding operations are inevitable and mandatory on a traditional computer system, decoherence of the quantum computer system would lead to a premature end of the computation on quantum systems.

Key words: Chebyshev, polynomial, real numbers, mantissa, Diffie-Hellman, RSA, key, exchange, polymorphic, encryption, cipher, asymmetric, symmetric, permutable polynomials, permutable rational functions, Shor’s algorithm, Grover’s algorithm, quantum, computer, qubit, decoherence, Heisenberg uncertainty, entangled, post-quantum.

1. Motivation

Key exchange algorithms and asymmetric encryption methods form the backbone of data exchange over the internet. The first and most widely used such algorithm was invented by Martin Hellman, Whitfield Diffie and Ralph Merkle at Stanford university in California, USA in the year 1976.

Widely used internet protocols such as SSL and TLS are based on the Diffie-Hellman or RSA key exchanges. It is technically impossible to solve certain algebraic operations in a finite field. For certain operations there exist solutions with exponential time complexity in the number of bits.

Since the invention of a ground-breaking factoring algorithm by Peter Shor [7] in 1994 and the (not proven) availability of quantum computer hardware capable of running the algorithm efficiently [1], [2], the scientific interest in finding a “Post-Quantum Key Exchange” - a key exchange that resists quantum computer attacks, is immense. Shor’s algorithm [7] is capable of factoring a number on quantum computer hardware with an execution time that is “only” polynomial in the number of bits. The operating principle was first verified experimentally in 2001 [8].

Quantum computers with approximately 512 qubits are now claimed to be available to companies like Lockheed Martin and government agencies like the CIA, NSA and NASA [1] [2]. This has fuelled speculations about the actual code-breaking capabilities of government organizations like the NSA. In order to break keys with 2,048 bit key length, approximately 6,150 qubits are although required. Quantum control and correction accounts for an even higher number of qubits that are required to mount an attack on 2,048 bit keys so that, depending on the chosen strategy, 20,000 .. 332,100 qubits must be present. The output of the quantum computer is not the desired solution. An additional step

¹ Gilbert Brands, D-26736 Krummhörn, e-mail: gilbert(at)gilbertbrands.de

² Bernd Röllgen, D-35576 Wetzlar, e-mail: roellgen(at)globaliptel.com

³ Kersten Vogel, CH-8055 Zürich, e-mail: kersten.vogel(at)sap.com

using conventional computers with time complexity $O(\text{bits}^3)$ is required to identify the actual key, provided the quantum system finishes in coherent state. Decoherence (change of state caused by spontaneous decay or interaction with the environment of a quantum system) drags the probability of the success of the quantum computation step down as the quantum lifetime is limited by experimental conditions, and eventually by the Heisenberg uncertainty principle in general. This problem increases with increasing number of bits used for the encryption because the number of quantum operations and, therefore, the computation time, increases with $O(\text{bits}^3)$ as well. The greater the size of the key that is attacked, the less the number of useful quantum computer results.

Finally there's the requirement for each qubit to interact with every other qubit multiple times which requires qubits to be brought into direct contact with each other. Alternatively, other entangled qubits could act as link but this would further increase the number of required qubits. Parallelization of operations is certainly achievable, but this inevitably leads to a greater number of erroneous signals [11].

Working systems in the sense of scientifically verified systems so far feature up to 14 entangled qubits [9], without executing operations that would be required to break ciphers. Measurements show that the stability of quantum systems drops rapidly when the number of qubits increases.

In 2015, IBM surprised with news [10] that could lead to a higher qubit count: they built a four qubit system using standard semiconductor chip construction techniques. A complete wafer would probably offer sufficient space for more circuitry, but the fundamental problem to make the circuits interact with each other seems to remain unsolved so far. Additional entangled qubits can possibly link distant qubits along a chain of such circuits as we assumed above, but this would require a completely different design.

Details about the alleged 512 qubit system [1][2] from D-Wave Systems could so far not be verified as these details are kept secret. In principle, the underlying technique (cooper pairs in superconductors) allows for the construction of adiabatic quantum computers. The difference to the above mentioned types is that in these systems the operator cannot control the transformations directly, and, therefore cannot implement an algorithm. Eve can only prepare the environment in a way that the desired solution will have the lowest energy level, and let the qubit system develop without further interaction hoping that it will reach this state. Quantum computers of this type are for this reason not universal because a special environment has to be developed for every problem (on a universal computer only the machine code needs to be changed). Theoreticians rate the underlying principle therefore as little promising for performing cryptanalysis or encryption operations.

Concluding from this short survey, working quantum computers that are able to attack today's encryption algorithms appear to be out of reach for quite some time. Furthermore does the fundamental lifetime limitation of quantum states defined by the Heisenberg uncertainty principle eventually limit the scalability of quantum computers. This limitation can be exploited by designing classical encryption algorithms with high complexity. The probability for a quantum system to generate a useful signal decreases exponentially with an increase in key length beyond a certain border. Even the NSA appears to think the same way as recommendations for key sizes appear to shift – very moderately, of course.

With the Heisenberg uncertainty principle in mind, quantum-proof key negotiation methods and encryption algorithms are of high scientific interest and there's a good chance for successfully developing such methods.

2. Prior art

Today's standard asymmetric key exchange and signature algorithms are based on algebraic functions over finite fields or rings. The well-known Diffie-Hellman and RSA algorithms, as a subclass, are based on simple commutative functions. Using a publicly known prime p , another publicly known parameter g and two secrets a and b belonging to Alice and Bob, a secret key can be negotiated by calculating:

$$\text{Bob: } \pi_a(g) \equiv g^a \pmod{p} \rightarrow \text{Alice}$$

$$\text{Alice: } \pi_b(g) \equiv g^b \pmod{p} \rightarrow \text{Bob}$$

$$\text{Both: } \pi_a(\pi_b(g)) \equiv (g^b)^a \equiv g^{a \cdot b} \equiv (g^a)^b \pmod{p} \equiv \pi_b(\pi_a(g)) \equiv \pi_{a \cdot b}(g)$$

This algorithm is vulnerable to QC (Quantum Computers) due to two properties: the algorithm has no variable parts, and the computation is performed over a finite field.

The general method can, however, be expanded to other commutative functions of this kind. Besides others, the so-called Chebyshev polynomials were investigated. Chebyshev polynomials, commonly abbreviated as “T polynomials”, are defined by the generators:

Recursive definition:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_n(x) = 2 \cdot x \cdot T_{n-1}(x) - T_{n-2}(x) \text{ for } n \geq 2$$

Analytical definition:

$$T_n(x) = \cos(n \cdot \arccos(x))$$

Iterative alternative (starting at the same origin as the recursive generator):

$$\begin{pmatrix} T_n(x) \\ T_{n+1}(x) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2x \end{pmatrix} \begin{pmatrix} T_{n-1}(x) \\ T_n(x) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2x \end{pmatrix}^n \begin{pmatrix} T_0(x) \\ T_1(x) \end{pmatrix}$$

These functions are primarily defined over the fields \mathbb{R}, \mathbb{C} with integer numbers n and variables in the definition range $[-1, 1]$ (for other subsets of the definition of some parts differ slightly, but this is of no importance for our task; the special properties and boundary conditions originate in special differential equations the polynomials are solution of). Chebyshev polynomials feature the commutative semi-group property in n :

$$T_a(T_b(x)) = T_{ab}(x) = T_b(T_a(x))$$

and may therefore be used in cryptography as well.

From Fig 1, and Fig 2 the reader may learn that

1. the interval $[-1, 1]$ is mapped to itself n -times by function $T_n(x)$
2. there exist a lot of cluster points where different functions meet
3. the mapping gets more dense with increasing n

We have to keep that in mind to evaluate the security of cryptographic algorithms.

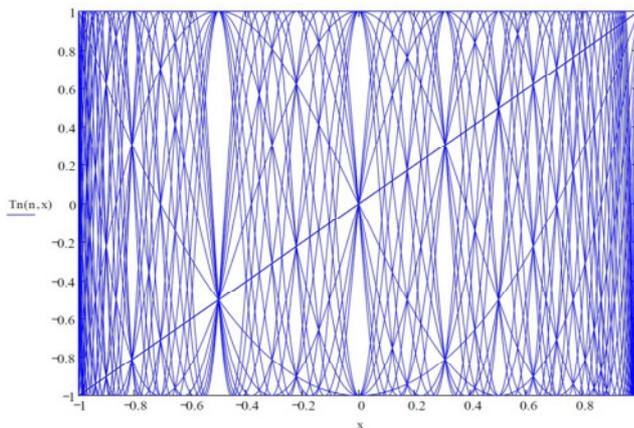


Fig. 1: Chebyshev polynomials $T_1(x)$.. $T_{20}(x)$

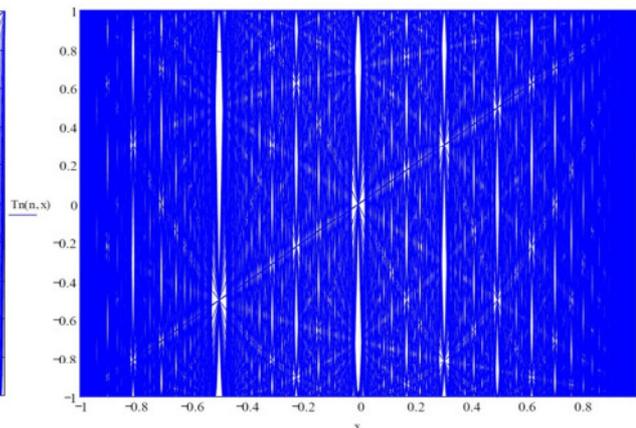


Fig. 2: Chebyshev polynomials $T_1(x)$.. $T_{128}(x)$

The recursive generator using T polynomials may be resistant to QC because the algorithm is no longer fix but variable because of the different sizes of the polynomials that are used. Using different values for n , Alice and Bob execute completely different computations. This is in contrast to executing the ordinary DH algorithm as such classical algorithms always use the same computation path regardless of the parameters.

Polymorphism poses an insurmountable hurdle for QC because quantum computers can simulate all different values at a time but not all different computation paths. So in a first investigation, T polynomials were applied over a finite field like DH, and not over the real number interval $[-1, 1]$. In [3] it was shown that this algorithm works but features no advantages over DH because it can be attacked with classical methods with the same complexity due to the iterative generator. QC attacks were, however, not investigated.

An algorithm using real numbers can also be constructed [5] using floating point operations on a classical computer. The authors claim such algorithms can be broken with less effort than brute force, but as will be shown later on, they completely missed the characteristics of numerics, and therefore their claim is false.

Algorithms using floating point computations were considered to be “evil” during a long time because the principles of numerical mathematics have to be observed. If mathematically accurate algorithms are exported to numerics, the associative, and the distributive law, and even in some environments the commutative law do not longer hold due to inaccuracies caused by rounding operations. On the other hand they are likely to be an ideal instrument to harden cryptographic algorithms against attacks using QC. Quantum computers operate by obeying to fundamental mathematical laws with the consequence that QC, just like classical computers with an inherently finite length of the mantissa, cannot process real numbers. Unlike actual implementations on classical computer hardware, which all perform rounding operations at the end of the mantissa, applying rounding rules to QC would lead to immediate decoherence because “rounding” would mean to assign new external values to qubits already present in the calculation which would destroy the entanglement of the system immediately.

3. Our approach

We decided to use T polynomials on real numbers and claim the algorithm is QC-proof because of two facts:

- 1.) The actual computation path differs strongly when varying the secret parameters
- 2.) QC is not suited to operate on real numbers.

The reader may argue that the T polynomials have a closed analytical generator, but implementations of the cosine and arccosine functions use a lot of different methods for different argument intervals including T polynomials themselves, so even the analytical generator is intrinsically polymorphic.

We will not completely preclude the possibility that someone may find a hack for this generator on quantum computer hardware in the future, but we think that a combination of two probably QC unsolvable methods gives room for the argument of QC safety, although we know this is not a mathematical proven theorem. It should as well be pointed out that a QC does not yet exist, which - as of today - renders QC algorithms intellectual dalliances.

The proposed key agreement algorithm works in analogy with the DH algorithm:

Alice and Bob agree on a public real number x , choose some large secret integer values r and s and compute:

$$T_r(x), T_s(x)$$

and subsequently exchange these values over an insecure channel.

The secret key is computed by combining the two functions

$$T_r(T_s(x)) = T_s(T_r(x)) = T_{s \cdot r}(x).$$

This method is in principle well known and assumed not to be secure [5] because an attacker Eve may compute

$$a(x) = \frac{\arccos(T_r(x))}{\arccos(x)}, \quad b(x) = \frac{\arccos(T_s(x))}{\arccos(x)}$$

using the analytical generator to yield two possible solutions which are related to r and s due to the periodicity of the cosine function. Looking at Fig. 1 we observe that Eve has found a T function which intersects with the functions used by Bob and Alice at the public argument x .

If Eve was in possession of Alice's and/or Bob's secret r and/or s , the following relationships could be exploited:

$$T_{rs}(x) = \cos(r \cdot b(x) \cdot \arccos(x)) = \cos(s \cdot a(x) \cdot \arccos(x))$$

Eve although can solely compute the parameters a and b at the public argument x , which is insufficient to compute the key (Alice's and/or Bob's shared secret):

$$T_{a,b}(x) = \cos(a \cdot b \cdot \arccos(x)) \neq T_{r,s}(x), \text{ and } T_{a(x),r}(x) \neq T_s(T_r(x))$$

because the intersection does hold for x , but not generally. Mathematically speaking, the commutative scheme of the T-polynomials only holds for integer factors because of their theoretical background, and therefore a real number $a(x)$ cannot be an attacker's solution.

In order to yield Alice's and Bob's secret key, Eve is required to run a number sieve. Due to the fact that the cosine function has a period of 2π , the complete set of solutions of the sieve is expressed by:

$$a_k(x) = \frac{\pm \arccos(T_s) + 2\pi \cdot k}{\arccos(x)} = \pm d + e \cdot k \quad \text{with } k \in \mathbb{Z}, \quad d, e \in \mathbb{R}$$

In order to reveal Alice's secret parameter s , Eve has to find a number k that solves the equation:

$$T_s(x) = \cos(s \cdot \arccos(x)) = \cos((\pm d + e \cdot k) \cdot \arccos(x))$$

In other words, a value k fulfilling the requirement $\pm d + e \cdot k \in \mathbb{N}$ is the target of the number sieve.

A solution for this problem is shown in [5], but as we remarked already, this solution misses completely the characteristics of numerics. The authors operated on secret values in the order of 100,000, but as can easily be verified using a pocket calculator, they only used the standard double precision floating point data type to derive their values. As can easily be verified experimentally, the usability of type double ends in the range of $T_{60}(x)$ because then the results contain nothing but rounding errors. The authors would have realized that if they would have calculated $T_s(T_r(x))$ in addition to $T_r(T_s(x))$, because in these "common secrets" of Alice and Bob not even the signs match. We therefore dispute these results, and claim that no useful solution exists as will be shown in the security section.

We will show the impracticality of a quantum computer attack first, and orientate ourselves at the available QC algorithm types. Provided Eve can develop a suitable mapping from floating point operations to a quantum system without falling into the trap of decoherence during rounding, Eve can use two QC approaches.

A first solution may be a direct search on a QC using Grover's algorithm [12]. But Grover's algorithm, directly looking for a solution, only reduces the complexity from $O(n)$ to $O(\sqrt{n})$. As long as n is within the range of 300 decimal digits, Grover's algorithm is no real alternative in order to find a solution.

Secondly Eve may also try to make use of a modified version of Shor's algorithm aiming for periodical events. Evaluating the above equation in the modular form $\pm d + e \cdot k \equiv 0(\text{mod}1.0)$, Eve may find $\pm d + e \cdot k' \equiv 0(\text{mod}1.0)$ where $k' = k + p$ with period p .

Substituting k with k' we find $\pm d + e \cdot k + e \cdot p \equiv 0(\text{mod}1.0)$ or $e \cdot p \equiv 0(\text{mod}1.0)$.

The attacker will only be able to identify p by Shor's algorithm but not k . Eve consequently cannot find the solution using the modified Shor's algorithm.

Further types of quantum computer algorithms are not in sight today. We therefore conclude that the proposed algorithm is QC safe since both known principles a QC may operate with to identify a solution, are not suited to solve the problem.

4. The technical solution

4.1 Strategies

We suggest at least three methods to realize a key agreement of pre-designed security. They can, however, be combined to satisfy even higher requirements on security on the cost of calculation time, and other resources.

It is obvious that T polynomials of high degree cannot be generated using the recursive or iterative method. The general approach of the first two methods to formally produce very high degrees is a combination of T polynomials, set up in the individual phases of Alice and Bob, to obtain very large secrets. From a pre-calculated set of T polynomials with moderate index, both peers select a number of these functions, and compute

$$f_{\text{Alice}} = T_a(T_b(T_c(\dots(x)))) = T_{a \cdot b \cdot c}(x)$$

$$f_{\text{Bob}} = T_d(T_e(T_f(\dots(x)))) = T_{d \cdot e \cdot f}(x)$$

As can easily be verified, the products increase very rapidly, yielding numbers with 200-600 decimal digits after only a few multiplications.

However, we need to observe that the resulting secret depends only on their factorization. Changing the order of the functions in the iteration doesn't change the result

$$T_a(\dots T_b \dots) = T_{a \cdot b} = T_b(\dots T_a \dots).$$

We have to take this into account when calculating the security. Above of this, different functions may yield the same secret due to common prime factors

$$T_9(\dots T_{17} \dots) = T_{153}(\dots) = T_3(\dots T_{51} \dots).$$

This can easily be taken into account by using only prime number indices in the function set.

4.1.1 Combination strategy

Alice and Bob each select N polynomials with arbitrary prime number indices p_1, p_2, \dots, p_N (the prime numbers may be chosen arbitrarily from a larger set of M numbers) and a maximum number of iterations (use of the very same T polynomial up to N times) w_1, w_2, \dots, w_N (these maximum numbers may be the same for all functions in practice, but may be individual in specialized implementations). The number of combinations s computes to yield:

$$s = \prod_{i=1}^N w_i \quad (= w^N \text{ if } w_i = w_j)$$

The actual number of iterations per T polynomial is selected independently by Alice and Bob $0 \leq v_i < w_i$. s is the primary security of the method. If for instance 128 functions are combined using $w = 2$, 2^{128} combinations are equally possible. The same number of equally probable combinations is yielded if the set of T polynomials only comprises 64 functions, but $w = 4$. The reader may easily conclude that this is only the lower limit of security, and the obtained security may be much higher than this because if $M > N$, Eve doesn't know the set of prime number indices that are in use, and if the w_i are chosen randomly, Eve doesn't know how to exactly conduct a brute force attack.

The highest possible number of convolutions, and therefore the quality of the secret in terms of the number of equally probable combinations is $d_{max} = \prod_{i=1}^N p_i^{w_i}$, while the actual number of convolutions is yielded by multiplying the prime number indices of the actually selected T polynomials:

$$r, s = d_v = \prod_{i=1}^N p_i^{v_i}.$$

These secret numbers outnumber the combinatorial magnitude by far as can easily be seen by conducting simulation experiments. For two different sets of functions and iteration parameters w , this may vary statistically in the range of 200 – 600 decimal digits, which approximately corresponds with 768 – 2.048 bits.

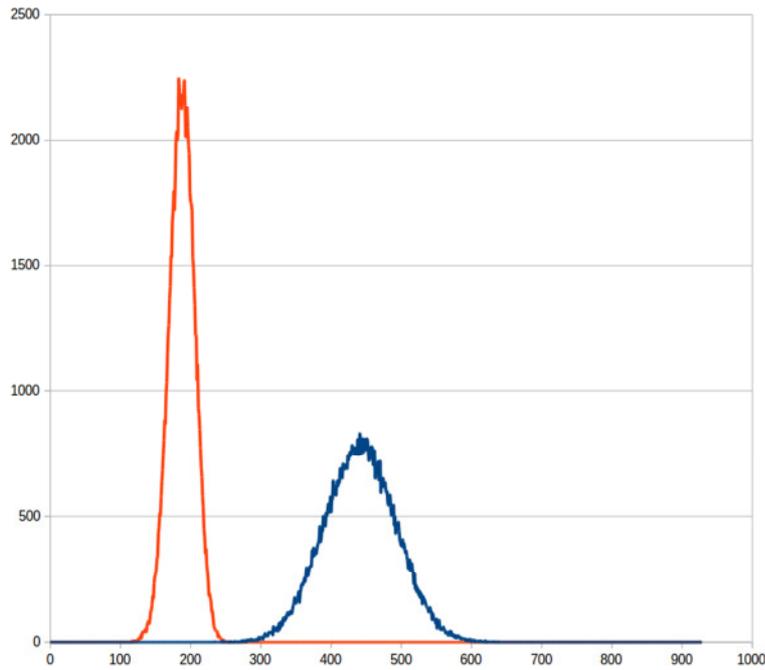


Fig. 3: Two combined T polynomials with approx. 128 bit security (64/4 and 128/2, functions/iterations). Abscissa: number of bits that are free from rounding errors, ordinate: actual number of convolutions.

Figure 3 shows that Alice and Bob can both influence the number of convolutions and the number of “useful” bits. The left graph shows a distribution function of secrets r, s resulting from selecting 64 T polynomials with small prime number indices and max 4 iterations yielding a maximum value of 10^{360} for the secrets and about 10^{180} as most probable values in arbitrary chosen iteration values. The right graph shows the same for a 128/2 distribution. We will show the impact on the method parameters and the resulting security in a later section of this paper.

Please observe:

- Every secret based on a set of functions and convolution values is unique,
- Alice and Bob need neither to agree on a certain function set nor on a repetition set making it hard for Eve to test whether a candidate for a secret r is composed of a certain set of primes.
- Eve has to find the exact secret. Even if Eve assumes ...08 instead of ...09 – a miss of the right most digit in a number containing 600 digits in total – a value that differs completely from the true $T_{r,s}$ is generated as can easily be verified experimentally.

To understand the reason for this effect, the reader should take into account that for a security of 128 bit, the magnitude of the secret needs to be at least in the order of 10^{39} , but the number of convolutions is in the order of magnitude $10^{200} - 10^{600}$. So between the smallest meaningful differences in the prime values, the combined T polynomials show at least 10^{160} extrema making it impossible to meet similar values.

4.1.2 Casket strategy

Instead of selecting a maximum numbers of iterations, Alice and Bob can combine the functions of the given set of magnitude n arbitrarily a given number of times r . There is no limit in how often a specific T polynomial in the set of available functions can be selected, and there is no guarantee that a certain polynomial is used at all.

There exist as many combinations as there are combinations with repetition while order doesn't matter in a classical experiment:

$$s = \binom{n+r-1}{r} = \frac{(n+r-1)!}{r!(n-1)!}$$

with n being the number of functions to choose from and r being the number of functions chosen.

The resulting number of convolutions or in other words the "magnitude of the secret" - can be calculated as above.

4.1.3 Analytical strategy

The analytical definition of T polynomials can be used to compute functions f_{Alice} and/or f_{Bob} in one step only. Alice and Bob choose a very large n of 200-600 digits arbitrarily and compute

$$T_n(x) = \cos(n \cdot \arccos(x))$$

n may be prime or not. Although this method seems to be an improvement, the analytical method is more time consuming to compute than iterative and recursive methods because the trigonometric functions rely on polynomials by themselves. The same precision of the floating point numbers can be assumed as for the other methods.

4.1.4 General remarks

Floating point precision. Since the common secret is based on $r \cdot s$, the floating point precision must take this into account. At least about 50 decimal digits at 128 bit security (for other security value chosen, the number of equal digits can be calculated accordingly - the reason for taking decimal digits instead of bits is explained in the next section) must be identical for Alice and Bob after finishing all computations. Using 64 functions with max 4 convolutions, the statistical values for the individual secrets are of magnitude in the range of 200 decimal digits, and therefore the common (unknown) product is of the order of 400 decimal digits. Increasing the formal degree of T polynomials increases the storage necessary to hold the coefficients in full precision by one bit per degree, and at least one bit precision per degree is lost due to rounding errors during calculation of the polynomial values. This results in very high precision of the floating point variables necessary to produce the desired amount of equal digits for Alice and Bob. Summing up: when using a floating point precision of 500 decimal digits in the above 128 bit 64/4 example, the number of „good decimal digits“ is less than 100. A total of 80% of the mantissa is lost due to rounding operations.

Security. The security originating from the number of combinations of 50 decimal digits seems to be somewhat outnumbered by the floating point accuracy of 500 decimal digits. It is possible to use the analytical method to work with smaller secret values, but we strongly recommend not to select small values for n in the analytical strategy to compensate for time efficiency losses. As the reader can deduct from figure 1 and figure 2, the function graphs deliver all but random values. The only chance to produce random values from arbitrary x is to use degrees of T polynomials that are far above the smallest rational difference in x which is 2^{-n} for a chosen minimum security of n bits. If $r \gg n$ there are many extrema of the function between two x values of minimal difference leading to function values taken from different edges of the graph, and therefore leading to more randomness.

Combination. The fastest method of calculation is the combination strategy, but using only functions with small prime indices might provide Eve with some sort of filter to test for candidates for secrets Eve might have retrieved from somewhere. In order to deprive Eve from utilizing such a filter, Alice and Bob may use the analytical method with large random integers or big prime numbers and pay with an increase in calculation time.

4.2 Technical aspects of implementations with conventional universal computers and hardware speedup

Recursive, iterative as well as analytical formulae for computing Chebyshev polynomials all entail use of long mantissae. Since there is a lot to do for the CPU, we favor C++ as the implementation language.

Many libraries with arbitrary length of the mantissa are available, so it is not necessary to implement long floats by yourself. To give room for optimization, nearly all libraries are opaque meaning the programmer of the security suite has no access to individual data bits. Instead of bit patterns, the most general interface are strings in decimal format. That's why most libraries allow to specify the length of the mantissa as number of decimal digits. We thus prefer to use the number of decimal digits in the discussion instead of binary digits. To allow for optimizing code, the number of digits has to be provided as a template parameter in C++. The disadvantage to this is that the length of the mantissa must be provided at compile time, which renders the task to switch between different precisions during runtime somewhat difficult.

The bit length of the mantissa needs to be at minimum as long as the length of the coefficients of the T polynomials. The length of these coefficients increases one bit per degree as can be seen from the generation formula. In addition to this, the same number of bits is lost during runtime because of rounding operations during the calculation of intermediate values. If Alice and Bob want to agree on a secret key of at least 128 bit security, at least the first 50 decimal digits of their common value must be identical. Putting it all together, the rule of thumb for the minimum number of decimal digits is $\log_{10}(d_{max})$. Figure 3 shows that the magnitude of secret values largely depends on the selected parameters. Starting from values derived from the rule of thumb, developers are encouraged to optimize the values experimentally.

The reader should observe that the precision of the floating point variables in the different methods must be of similar size if the size of the secrets is in a similar range. One might think that the analytical formula might get along with less precision, but this is not the case. In order to calculate $\cos(r \cdot \arccos(x))$ with a large r , the operation results in $\cos((r \pmod{\pi}) \cdot \arccos(x))$, and the modulus itself is a subtraction of large numbers yielding a result which is relatively small compared with the initial values, and therefore erases digits in the order of r itself (cf. our remarks on [5]).

There are different possibilities to choose the working parameters, starting with the definition of the initial combination security which can be obtained by selecting different numbers of functions and convolutions. For instance, 32 functions with max. 8 iterations, 64 T polynomials with 4 iterations, or 128 functions with 2 iterations all result in 2^{128} possible states. All these possibilities lead to different d_{max} values, and to different precisions therefore. The chosen values may differ due to personal ideas of the resulting security. The developer should have in mind that storage and time consumption heavily depends on these parameters. The relative storage consumption can be calculated from:

$$M = \sum_{i=1}^N p_i \cdot precision$$

The relative time consumption of a function set yields:

$$T \approx precision^a * \sum_{i=1}^N p_i \cdot w_i \quad ; \quad 1.4 < a \leq 2$$

The power of the precision parameter in the formula depends on the degree of optimization of the math library, and may range between 2 (no optimization) and 1.4 (FFT multiplication). For instance, 64/4 consumes about 1.2 MB of storage, 128/2 up to 5.8 MB (500 digits), 256/4 finishes at 108 MB and 2300 digits, but on the other hand resulting in 512 bit security. The time consumption on a standard PC varies from 160 ms over 630 ms to 49.8 seconds.

The reader may notice that parameter selection allows for a great degree of freedom. We therefore encourage developers to identify suitable parameters (lower range of precision, function/repetition, etc.) by experiment. Meeting today's standard security considerations, the algorithm can easily compete with Diffie-Hellman if the parameters are chosen carefully. If a user prefers extraordinary security parameters, the algorithm can be extended arbitrarily at the cost of memory and calculation time.

In application software, time consumption may be the primary parameter to be optimized. The programmer may choose less significant digits in the floating point types than are necessary to guarantee identical common secrets under all circumstances. The parameters may as well be configured in a way to get to a common secret in 98% of all cases. The protocol has to take this 2% failure rate into account and resume the key negotiation with another parameter set.

When implementing the proposed key exchange into a key exchange protocol, care needs to be taken in the following cases:

- Function outputs (-1, 0, 1) must be avoided because these values are mapped into (-1, 0, 1) by any T polynomial.
- $x = T_a(x)$ is only problematic if this is the result of the combined function of T polynomials for Alice or Bob. The effect can be countered by adding or removing an iteration, or by modifying prime number coefficients, or by just selecting another x .

These events might never occur in reality due to the enormous number range in which the key exchange operates.

5. Security

There exist two methods that potentially allow to break the proposed polymorphic key exchange: Eve can either try to solve the linear equation

$$a_k = \frac{\pm \arccos(T_r) + 2 \cdot \pi \cdot k}{\arccos(x)} = \pm d + e \cdot k \quad \text{with } k \in \mathbb{Z} \quad \text{and } d, e \in \mathbb{R},$$

or Eve can try to identify the exact combination of T polynomials. There exists no other interrelation that can be used to mount an attack. The only two relationships that Eve can possibly take advantage of are discussed in 5.1 and 5.2.

5.1 Brute force sieve over the product of prime number indices

As already described earlier, any attack on Alice's and/or Bob's secret r and s entails the necessity to cope with the periodicity and the symmetry of the cosine function. Eve has to find correct integer parameters solving

$$a_k = \frac{\pm \arccos(T_r) + 2 \cdot \pi \cdot k}{\arccos(x)} = \pm d + e \cdot k \quad \text{with } k, a_k \in \mathbb{Z} \quad \text{and } d, e \in \mathbb{R}.$$

in order to reveal the secret key. To arrive there, Eve may transform the equation by means of some factors M to a linear modular equation

$$k \cdot [e \cdot M] \equiv [\pm d \cdot M] \pmod{M}$$

where $[..]$ denotes the rounding of the enclosed product to the nearest integer value. Equations of this kind are called diophantine equations, and can easily be solved by well-known methods as is shown in [5].

The solutions of diophantine equations are at most of the order of M . To break into the „security“, Eve must guess the correct value precisely. As can be easily shown by experiments, Eve doesn't yield the key if she misses the exact number of convolutions, which is in the of magnitude of 10^{300} , by +/- 1 decimal digit! Since Eve must guess the correct value, she has to provide a sufficiently large M . The problem however is the rounding operation $[..]$. There are at least twice as much exact digits in a correctly configured key agreement suite, so Eve must not fear to miss information from rounding, but the integer values she uses are not exact. The digits cut by the rounding to integers would propagate to the highest-order decimal digits in an exact calculation yielding $\pm d + k * e \notin \mathbb{Z}$, and therefore k is not a solution of the problem. If Eve manages to arrive at 1234... , and the exact value starts with 1235..., the search space of Eve contains 10^{177} values in case of a 64/4 encryption. Our investigations show that Eve doesn't even come this close to the correct solution. Starting sequences which differ as early as in the first digit are more likely. Our investigations show further that the estimated k increases with M and doesn't stop at a certain length when M exceeds the magnitude of the secret. When selecting $M \gg r$, the solution of the diophantine equation becomes $\pm d + k * e \gg r$. Since the order of the secret may be 180 +/- 20 digits, Eve likewise doesn't even find a manageable interval to conduct an exhaustive search in.

To sum it up, we claim that due to the properties of numeric systems, Eve has no chance to break into the secrets by other methods than brute force. The complexity of attacking the product of prime number indices may be larger than directly trying to attack the combination of T polynomials.

5.2 Brute force sieve over the entirety of combinations of T polynomials

The calculated security depends on the combinations of functions and the number of convolutions. As we already mentioned, the effective security may even be much higher without touching these parameters by choosing the prime indices randomly instead of consecutively, or by choosing the maximum repetition values individually. So a formal 128 bit security may easily be as complex as 200 bit or more for Eve.

From a theoretical point of view we cannot preclude that more than one combination of the functions produce the same intermediate public value with parameter x within the number of precise digits, although this seems to be unlikely in practice. This will be of no impact on the common secret because if this happens, Eve has found another polynomial intersecting Alice's combination of T polynomials at x as this is the case for the inversion of the analytical formula. But this will surely not be the case for other values, especially not for calculating the common secret with Bob's intermediate value. So Eve can only break the secret key if she finds the exact combination of Alice's or Bob's functions.

Using the analytical formula with an arbitrary large r of > 100 digits may even lead to a larger computational security, but we remind the reader of the remarks in section 4.1.1, that there must be a large number of extrema between the smallest meaningful difference of x values to guarantee cryptographic secure randomness.

5.3 Randomness of the bit patterns of exchanged keys

Alice and Bob's shared key $T_{s,r}(x)$ has been tested experimentally for randomness with arbitrary values for the parameters and x as well as with chosen values with small known differences between the values using the Diehard battery of randomness tests [13].

Due to the nature of Chebyshev polynomials, namely that the slope at y -values near 0 is at its maximum, the leading digits are not uniformly distributed. This property can be noticed in the following tests:

- Overlapping-Pairs-Sparse-Occupancy: digits 1 .. 10 and 23 .. 32
- COUNT-THE-1's TEST for specific bytes: digits 1 .. 8, 2.. 9, 3 .. 10
- SQUEEZE test: p-value=1.000000
- CRAPS TEST: p-value for no. of wins: .002047, p-value for throws/game: .992118

It can easily be seen that the three high-order digits are not evenly distributed - as expected when looking at the function graphs.

All other tests were passed without the slightest deviation from perfect randomness. The positive test result leads to the conclusion that the quality of Alice's selection of a random x can feature imperfect randomness without affecting security of the key exchange.

The agreed secret value can be transformed to a secret encryption key by hashing an appropriate number of decimal digits. For a security of 128 bit, the decimal number of possibilities is $3.4 \cdot 10^{38}$, for 256 bit $1.2 \cdot 10^{77}$. Taking into account the deviation from randomness of the first digits, 50 respectively 90 digits are sufficient to agree on a secret key for, say, AES, or Twofish after compression with an appropriate hash function transforming human readable decimal digits into a suitable number of binary digits for the symmetric cipher.

This minimum number of digits that are free from rounding errors, is a necessity for the key exchange to succeed. As

the developer is advised to optimize the parameters experimentally, the key agreement may not succeed with certain parameter sets. The protocol has to take this into account, for instance by exchanging a hash value generated by a separate hash function to test for equality, and a resume operation if a key exchange was not successful.

5.5 The need for high quality true randomness

An attacker generally targets the soft spot in a system. Therefore it is of no value if the algorithm is configured to yield 512 bit of security, consumes hundreds of megabytes of RAM and takes 50 seconds to execute, while other parts of the security suite are not of the same quality.

To start at the end, there is no reason to agree on a 256 bit key if AES-128 is used, and even AES-256 may not result in 256 bit security if pure ECB mode is used instead of chaining modes. Key agreement and succeeding symmetric algorithm should match in their security parameters, and both security components should match the importance of the data. A high security key of 256 tera years of resistance against attacks is rubbish if someone wants to say “hello” to his friend, or if this data does not remain secret for the next three days.

At the other end, a random generator of appropriate quality is needed. The random number generators that are typically built into standard libraries are designed to help solving mathematical problems, and are not intended to be used for high-end cryptography! At best these implementations are initialized using system time, but as a year contains only $3.15 \cdot 10^{10}$ milliseconds, and as an attacker may guess the start time of an application, Eve may simulate all possibilities in a few hours, and guess the correct secret values. Cryptographically secure random generators are built on top of hash functions, or ECC, and run much slower. On the other hand do such RNGs output random number sequences of arbitrary security if properly re-seeded with true random data.

6. Source Code

C++ source code demonstrating the numerical part of the proposed key exchange is available at:

http://downloads.globaliptel.com/cryptography/RVB_key_exchange/

and

<http://www.gilbertbrands.de/neustart/>

References:

- [1] Tom Simonite. The CIA and Jeff Bezos Bet on Quantum Computing. Article: MIT Technology Review. <http://www.technologyreview.com/news/429429/the-cia-and-jeff-bezos-bet-on-quantum-computing/>, 2012
- [2] Alex Mansfield. Nasa buys into 'quantum' computer. Article: BBC News Science & Environment. <http://www.bbc.com/news/science-environment-22554494>, 2013
- [3] G.J. Fee, M.B. Monagan, Cryptography using Chebyshev polynomials. <http://www.cecm.sfu.ca/CAG/papers/Cheb.pdf>, 2003
- [4] Garry J. Tee, Permutable Polynomials and Rational Functions, http://nzjm.math.auckland.ac.nz/images/3/31/Permutable_Polynomials_and_Rational_Functions.pdf, 2011
- [5] P. Bergamo, P. D'Arco, A. De Santis, L. Kocarev. Security of Public Key Cryptosystems based on Chebyshev Polynomials. http://www.di.unisa.it/~paodar/preprint/pdf/web_chaos.pdf, 2004
- [6] L. Kocarev, Z. Tasev, P. Amato, Rizzotto, Encryption process employing chaotic maps and digital signature process, 2005, US Patent 6,892,940 D2 (Foreign Application Priority: Apr. 7, 2003, EP 03425219)
- [7] Shor's Factoring Algorithm, Notes from Lecture 9 of Berkeley CS 294-2, 4 Oct 2004, <http://www.cs.berkeley.edu/~vazirani/f04quantum/notes/lec9.ps>
- [8] IBM's Test-Tube Quantum Computer Makes History, Press release, 19 Dec 2001, <http://www-03.ibm.com/press/us/en/pressrelease/965.wss>
- [9] T. Monz, P. Schindler, J. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Haensel, M. Hennrich, R. Blatt, 14-Qubit Entanglement: Creation and Coherence, 2011, <http://mina4-49.mc2.chalmers.se/~gojo71/KvantInfo/LiteratureProjectPapers/14-Qubit%20Entanglement%20Creation%20and%20Coherence.pdf>
- [10] IBM Scientists Achieve Critical Steps to Building First Practical Quantum Computer, Press release, 29 Apr 2015, <http://www-03.ibm.com/press/us/en/pressrelease/46725.wss>
- [11] G. Brands, Einführung in die Quanteninformatik, 2011, Springer Verlag, ISBN 978-3-642-20646-7
- [12] L. K. Grover, A fast quantum mechanical algorithm for database search, 1996, <http://arxiv.org/pdf/quant-ph/9605043v3.pdf>
- [13] G. Marsaglia, Website: The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, <http://www.stat.fsu.edu/pub/diehard/>