KEY EXCHANGE USING CHEBYCHEV POLYNOMIALS

M.D. Atkinson and Vincenzo Acciaro

SCS-TR-165, JANUARY 1990

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

Key exchange using Chebychev polynomials

M.D. Atkinson
Vincenzo Acciaro

School of Computer Science
Carleton University
Ottawa, CANADA K1S 5B6

1. Introduction

The well-known Diffie-Hellman protocol for key exchange [Dif76] allows two parties to agree on a secret key without the key being sent along an insecure communication channel. This protocol is a special case of the following general scheme which allows two parties A and B to agree on an element of a key space K.

Let $\boldsymbol{F} = \{F_m | \ m=0,1,2,....\}$ and $\boldsymbol{G} = \{G_n | \ n=0,1,2,....\}$ be two families of functions defined on K having the property that for all m, n we have the equality $F_m(F_n(x))=F_n(F_m(x))$. Initially A and B agree on an element $\alpha \in K$ which may be made public. Next, A chooses a private integer m and sends $F_m(\alpha)$ to B while B chooses a private integer n and sends $G_n(\alpha)$ to A. Finally A computes $F_m(G_n(\alpha))$ while B computes $G_n(F_m(\alpha))$; by the assumption on $\boldsymbol{F}$ and $\boldsymbol{G}$ these two results are equal and serve as their common key.

In all the examples given in this paper $\boldsymbol{F} = \boldsymbol{G}$ and we shall assume this from now on; it would however be interesting to have examples where $\boldsymbol{F} \neq \boldsymbol{G}$. In the original Diffie-Hellman scheme $\boldsymbol{F}$ and $\boldsymbol{G}$ were each the set of functions $\{x \rightarrow x^m \bmod p, m=0,1,2,3...\}$ (where p is prime) and K was the field Z_p of integers modulo p. In [Rue88] $\boldsymbol{F}$ and $\boldsymbol{G}$ consisted of the functional powers of some function p(x). The condition that members of $\boldsymbol{F}$ commute with one another under functional composition is a very restrictive one. However there is another family of polynomials which commute under composition:- the Chebychev polynomials. Indeed it is readily seen that, if $T_n(x) = \cos(m \arccos x)$, we have

$$T_m(T_n(x)) = \cos(m \ \arccos (\cos (n \ \arccos x)) = \cos(mn \ \arccos x) = T_{mn}(x)$$

A basic recurrence, following from properties of the cosine function, relating the Chebychev polynomials is

$$T_{n+1}(\alpha) - 2\alpha \, T_n(\alpha) + T_{n-1}(\alpha) = 0$$

Since the Chebychev polynomials have integer coefficients we may regard them as polynomials defined on Z_p and the above equations still hold (and from now on we shall use arithmetic modulo p). In order for the Chebychev polynomials to yield a *practical* method for key exchange it is essential that $T_n(\alpha)$ be computable rapidly, that α be chosen so that there is a large collection of potential keys which may be generated by the algorithm, and that it should not be feasible to deduce n given the value of $T_n(\alpha)$ (which can be intercepted by an enemy during the key exchange protocol).

2. Efficient computation of $T_n(\alpha)$

We shall give three methods for computing $T_n(\alpha)$ all of which require O(log n) arithmetic operations in Z_p. In the first method we express the recurrence above in matrix notation as

$$\begin{pmatrix} T_n(\alpha) \\ T_{n+1}(\alpha) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2\alpha \end{pmatrix} \begin{pmatrix} T_{n-1}(\alpha) \\ T_n(\alpha) \end{pmatrix} = U \begin{pmatrix} T_{n-1}(\alpha) \\ T_n(\alpha) \end{pmatrix}$$

Clearly, since $T_0(\alpha)=1$ and $T_1(\alpha)=\alpha$,

$$\begin{pmatrix} T_n(\alpha) \\ T_{n+1}(\alpha) \end{pmatrix} = U^n \begin{pmatrix} 1 \\ \alpha \end{pmatrix}$$

Since U^n can be computed in O(log n) steps by the binary method $T_n(\alpha)$ can be computed in this number of steps also.

The second method uses the explicit solution of the recurrence:

$$T_n(\alpha) = \frac{1}{2}(\lambda^n + \lambda^{-n})$$

where λ and λ^{-1} are the roots of $\lambda^2 - 2\alpha\lambda + 1 = 0$. The quantity λ may not lie in Z_p but, in any case, its powers may be computed by the binary method and thus $T_n(\alpha)$ can be computed in O(log n) steps.

These two techniques are generally applicable to any sequence defined by a linear recurrence. Our third method uses the special properties of Chebychev polynomials and is very simple to implement. It may be expressed by the recursive rules:

$$T_0(\alpha) = 1$$

$$T_1(\alpha) = \alpha$$

$$T_2(\alpha) = 2\alpha^2 - 1$$

$$T_n(\alpha) = T_{n/2}(T_2(\alpha)) \text{ if n is even and } n>2$$

$$T_n(\alpha) = \frac{T_{n-1}(\alpha) + T_{n+1}(\alpha)}{2\alpha} \text{ otherwise}$$

For simplicity these formulae ignore the possibility that $\alpha = 0$ but this is easily handled since $T_n(0) = 0$ when n is odd and $T_n(0) = (-1)^{n/2}$ when n is even.

Notice that, in the last recursive rule, each of n-1 and n+1 are even and one of them is a multiple of 4. It follows that $T_n(\alpha)$ may be computed in O(log n) steps.

3. Inverse computation

In this section we study the inverse problem of computing n given $T_n(\alpha)$. If there should be an efficient algorithm for this then the key exchange protocol would be worthless. But it is easy to see that the problem is of a difficulty comparable with computing a discrete logarithm. From the explicit solution to the recurrence the inverse problem is that of computing n given the value y in the equation

$$y = \frac{\lambda^n + \lambda^{-n}}{2}$$

But this equation gives a quadratic equation for λ^n and once λ^n is known n itself can be calculated by taking a discrete logarithm. Conversely a method for solving this equation induces a method for solving $\lambda^n = z$ (we solve $\frac{\lambda^n + \lambda^{-n}}{2} = \frac{z+z^{-1}}{2}$).

The above remarks include a specific method for solving the inverse problem. As we shall see shortly there is also a direct solution along the lines of the Pohlig-Hellman algorithm [Poh78] for computing discrete logarithms but, before discussing this, it is necessary to make a few remarks on the values assumed by $T_n(\alpha)$, n=0,1,2,....

Since $\lambda=\alpha + \sqrt{\alpha^2 - 1}$, either $\lambda \in Z_p$ or λ lies in the quadratic extension of order p^2 depending on whether $\alpha^2 - 1$ is a quadratic residue modulo p.

Lemma 1 For all choices of $0\neq\alpha \in Z_p$, the corresponding λ has order dividing p-1 or p+1. Moreover, if λ is chosen to have order dividing p-1 or p+1 or then $\alpha = \frac{\lambda+\lambda^{-1}}{2} \in Z_p$.

Proof. If $\alpha = \frac{\lambda+\lambda^{-1}}{2} \in Z_p$, $\lambda+\lambda^{-1} = (\lambda+\lambda^{-1})^p = \lambda^p + \lambda^{-p}$, which is equivalent to the relation $\lambda^{-p}(\lambda^{p-1} - 1)(\lambda^{p+1} - 1) = 0$.

It is evident from this lemma that the sequence $T_n(\alpha)$, n=0,1,2,.... is periodic with period dividing p-1 or p+1 according as α^2-1 is or is not a square modulo p. If the period is short then the protocol will be incapable of generating a large number of keys and the resulting crypto-system will be vulnerable to attack by systematic key trials. On the other hand, if λ has order p-1 or p+1, then, as the next result shows, about half of the elements of Z_p can arise as potential keys almost all with equal probability.

Lemma 2 (i) If λ has order p-1 then ($T_n(\alpha)$, n=0,1,2,...) has period p-1 and the set $\{T_n(\alpha), n=0,1,2,...p-2\}$ has size (p+1)/2.

(ii) If λ has order p+1 then ($T_n(\alpha)$, n=0,1,2,...) has period p+1 and the set $\{T_n(\alpha), n=0,1,2,...p\}$ has size (p+3)/2.

Proof. In either case, if $z \in Z_p$, the equation $T_n(\alpha)=z$ may be written $\lambda^{2n} - 2z\lambda^n + 1 = 0$. If $z=\pm 1$ this has one solution for λ^n and otherwise it has 0 or 2 solutions for λ^n (in the latter case one solution is the inverse of the other). The value of λ^n determines n modulo the order of λ. In case (i) the sequence has period dividing p-1. Among the first p-1 terms each value of $T_n(\alpha)$ except for the two which correspond to $\lambda^n=1$ and to $\lambda^n=-1$ occurs exactly twice, and so the set $\{T_n(\alpha), n=0,1,2,...p-2\}$ has size 2+(p-3)/2=(p+1)/2. Since the set is of size greater than (p-1)/2 the period is exactly p-1. Case (ii) is similar.

Suppose we wish to find the values of n for which $T_n(\alpha)=y$ in the cases that λ has order p-1 or p+1. The case are similar so we suppose that λ has order p-1. The following method will be effective if p-1 is divisible by small primes only. Let $p-1 = \prod_{i=1}^{k} p_i^{e_i}$. We shall determine n mod $p_i^{e_i}$ for each i and then use the chinese remainder theorem. Suppose that $v = n \bmod p_i^{j-1}$ is known (initially we can take j=1). Write $n = qp_i^j + up_i^{j-1} + v$, with $0 \le u < p$. If we can find u we shall know n mod p_i^j and the process can be repeated.

We compute
$$z = T_{(p-1)/p_i^j}(y) = T_{n(p-1)/p_i^j}(\alpha)$$
$$= T_{(p-1)q+(p-1)u/p_i+v(p-1)/p_i^j} = T_{(p-1)u/p_i+v(p-1)/p_i^j}.$$

Then we find u by computing the right hand side for each of the p possibilities for u in this equation.

In the Diffie-Hellman key exchange method it is generally recommended that the prime p should be chosen so that p-1 is not the product of small primes only; then the protocol cannot be compromised by the Pohlig-Hellman algorithm. For the scheme above a less restrictive condition on p applies:- one of p-1 and p+1 must not be a product of small primes only.

References

[Dif76]. Whitfield Diffie, Martin E. Hellman, New directions in cryptography, IEEE Trans. on Information Theory, IT-22 (1976), 644-654.

[Poh78]. Stephen C. Pohlig, Martin E. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Trans. on Information Theory, IT-24 (1978), 106–110.

[Rue88]. Rainer A. Rueppel, Key agreements based on functional composition. Advances in Cryptology - Eurocrypt 1988 (Editor Christoph G. Günther), Lecture Notes in Computer Science 330, pp3-10, Springer-Verlag (Berlin-Heidelberg-New York-London-Paris-Tokyo).

RECENT REPORTS

SCS-TR-147 **On Transparently Modifying Users' Query Distributions**
B.J. Oommen and D.T.H. Ng, November 1988

CS-TR-148 **An O(N Log N) Algorithm for Computing a Link Center in a Simple Polygon**
H.N. Djidjev, A. Lingas and J.-R. Sack, July 1988
Available in STACS 89, 6th Annual Symposium on Theoretical Aspects of Computer Science, Paderborn, FRG, February 16-18, 1989, Lecture Notes in Computer Science, Springer-Verlag No. 349

SCS-TR-149 **Smallscript: A User Programmable Framework Based on Smalltalk and Postscript**
Kevin Haaland and Dave Thomas, November 1988

SCS-TR-150 **A General Design Methodology for Dictionary Machines**
Frank Dehne and Nicola Santoro, February 1989

SCS-TR-151 **On Doubly Linked List ReOrganizing Heuristics**
D.T.H. Ng and B. John Oommen, February 1989

SCS-TR-152 **Implementing Data Structures on a Hypercube Multiprocessor, and Applications in Parallel Computational Geometry**
Frank Dehne and Andrew Rau-Chaplin, March 1989

SCS-TR-153 **The Use of Chi-Squared Statistics in Determining Dependence Trees**
R.S. Valiveti and B.J. Oommen, March 1989

SCS-TR-154 **Ideal List Organization for Stationary Environments**
B. John Oommen and David T.H. Ng, March 1989

SCS-TR-155 **Hot-Spot Contention in Binary Hypercube Networks**
Sivarama P. Dandamudi and Derek L. Eager, April 89

SCS-TR-156 **Some Issues in Hierarchical Interconnection Network Design**
Sivarama P. Dandamudi and Derek L. Eager, April 1989

SCS-TR-157 **Discretized Pursuit Linear Reward-Inaction Automata**
B.J. Oommen and Joseph K. Lanctot, April 1989

SCS-TR-158 **Parallel Fractional Cascading on a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, May 1989

SCS-TR-159 **Epsilon-Optimal Stubborn Learning Mechanisms**
J.P.R. Christensen and B.J. Oommen, June 1989

SCS-TR-160 **Disassembling Two-Dimensional Composite Parts Via Translations**
Doron Nussbaum and Jörg-R. Sack, June 1989

SCS-TR-161 **On the Data Analysis of Random Permutations and its Application to Source Recognition**
R.S. Valiveti and B.J. Oommen, June 1989

SCS-TR-162 **An Adaptive Learning Solution to the Keyboard Optimization Problem**
B.J. Oommen, R.S. Valiveti and J. Zgierski, October 1989

SCS-TR-163 **Finding a Central Link Segment of a Simple Polygon in O(N Log N) Time**
L.G. Alexandrov, H.N. Djidjev, J.-R. Sack, October 1989

SCS-TR-164 **A Survey of Algorithms for Handling Permutation Groups**
M.D. Atkinson, January 1990

SCS-TR-165 **Key Exchange Using Chebychev Polynomials**
M.D. Atkinson and Vincenzo Acciaro, January 1990