How to Exploit the Intractability of Exact TSP for Cryptography

Stefan Lucks

Institut für Numerische und Angewandte Mathematik
Georg–August–Universität Göttingen
Lotzestr. 16–18, D–37083 Göttingen, Germany
(email: `lucks@namu01.gwdg.de`)

Abstract. We outline constructions for both pseudo-random generators and one-way hash functions. These constructions are based on the exact TSP (XTSP), a special variant of the well known traveling salesperson problem. We prove that these constructions are secure if the XTSP is infeasible. Our constructions are easy to implement, appear to be fast, but require a large amount of memory.

1 Introduction

In some fields of modern cryptography, e.g. public key cryptography, it is common to base the security of cryptosystems on the intractability of well known mathematical problems. Examples for these problems are the factorization of integers and the discrete logarithm. But for the fields of one-way hash functions and secret key cryptography there seems to be no scheme, which has a simple mathematical description, is provably secure under a reasonable assumption *and* is fast.

Merkle and Hellman [8] were the first to suggest an NP-hard problem for cryptography. They couldn't prove their public key cipher to be as secure as the underlying problem, and later their cryptosystem was broken. In fact it is regarded to be very unlikely, that one can prove the equivalence of breaking a public key cipher and computing some NP-hard problem.

Shamir [13] suggested an identification scheme based on the NP-hard permuted kernel problem (PKP). For a discussion of the hardness of the PKP see [1], [5] and [11]. Other identification schemes based on NP-hard problems were due to Stern, see [14] and [15].

What about secret key cryptography? Theoretical constructions are known which are as hard to break as any one-way function, though these constructions are too inefficient for practical applications. Furthermore Impagliazzo and Naor [6] did discuss constructions for pseudo-random bit generators and universal one-way hash functions, which are as secure as the Subset Sum problem. This is NP-hard.

A one-way hash function very similar to Impagliazzo's and Naor's scheme was suggested the same year by Damgård [4]. This was broken by Camion and Patarin [2], using essentially brute force and applying the birthday paradox. We conclude that Damgård's scheme did not fall due to an inherent feasibility of the Subset Sum problem.

Anyway the Subset Sum problem *may* be too easy for cryptography. The whole theory of NP-hardness is based on worst-case complexity. Therefore NP≠P would only imply that Subset Sum is infeasible in the worst case. But even being infeasible on average might be not enough.

Consider a problem P and assume for simplicity, that all instances of size n are equally likely. Let P be hard on average. In other words, the expected runtime for every probabilistic algorithm to solve P, when an instance of size n is randomly chosen, is exponential (in n). There may exist an algorithm, such that, say, for 50% of all instances of size n we need exponential time to solve them, and for the remaining instances we need time n^c. The expected runtime for this algorithm is exponential, but a randomly chosen instance is easy with probability 1/2.

In cryptography we demand for any attacker the probability of a successful attack to be negligible. Consequently we also demand that a randomly generated instance of the underlying problem is hard with overwhelming probability—not only hard in the worst case or hard on average.

Empirical results on heuristics for the Subset Sum problem—e.g. by Schnorr and Euchner [12]—raise serious doubts on the security of Impagliazzo's and Naor's schemes. Nevertheless other NP-complete problems can be good cryptographic one-way functions.

The traveling salesperson problem (TSP) is among the oldest and most prominent problems in algorithm and computational complexity theory. It is unsolved if we only regard an efficient algorithm as a valid solution. It has been studied long before the theory of NP-hardness was developed, see [7] for details.

2 The Exact TSP (XTSP)

Essentially the XTSP is a variant of the TSP, where we are looking for a Hamiltonian path of a given length—not for the shortest one.

In the following $A = (a_{i,j})$ is an $n \times n$-matrix with $a_{i,i} = 0$ and for $i \neq j$ $a_{i,j}$ randomly chosen from $\{0, \ldots, 2^{l(n)} - 1\}$. We think of A as the distance matrix for distances in the complete directed Graph G_n with n vertices. Therefore the XTSP is actually a family of problems depending on the parameter $l(n)$.

In this paper we only deal with directed graphs, but our results can easily be adopted to the undirected case too.

We regard the numbers $a_{i,j}$ as fixed public constants like, say, the S-boxes of DES.

Any Hamiltonian cycle X for G_n can be coded as an integer with

$$\lceil \log_2((n-1)!) \rceil \quad \text{Bits.}$$

By $\text{Length}_A(X)$ we mean the length of X with respect to A. Given a number B, the XTSP is to find a Hamiltonian cycle X with

$$\text{Length}_A(X) = B.$$

It is easy to prove the NP-hardness of the XTSP and the NP-completeness of the corresponding existence problem.

By the following theorem we find a relationship between different members of our problem family.

Theorem 1. *Let*

$$l'(n) \ll l(n) \ll \log_2((n-1)!) \quad or \quad l'(n) \gg l(n) \gg \log_2((n-1)!).$$

Then the XTSP with number length $l'(n)$ can—with respect to probabilistic algorithms and except for a polynomial factor in computation time—be no harder than the XTSP with $l(n)$.

Sketch of proof: For $l'(n) \ll l(n) \ll \log_2((n-1)!)$ we regard the difference matrices $A = (a_{i,j})$ and $A' = (a'_{i,j})$ with $a'_{i,j} \equiv a_{i,j} \pmod{2^{l'(n)}}$. For any random X and $B = \mathrm{Length}_{A'}(X)$ there exists with overwhelming probability a Y with $\mathrm{Length}_A(Y) = B$. If Y exists, it is a solution with respect to A' too.

The proof for $l'(n) \gg l(n) \gg \log_2((n-1)!)$ uses the fact, that any solution with respect to the number length $l(n)$ is unique with overwhelming probability. Then any solution for the number length $l'(n)$ is—if existing at all—the same as for $l(n)$. □

Thus $l(n) \approx \log_2((n-1)!)$ describes the most secure cases. It seems recommendable to bound $l(n)$ by $\frac{1}{c}\log_2((n-1)!) \leq l(n) \leq c\log_2((n-1)!)$ for some $c > 1$, e.g. $c = 2$.

To get a "more uniform" output we define the modular XTSP. Given the matrix A, the number B and the number length $l(n)$, the problem is to find an X with

$$\mathrm{Length}_A(X) \equiv B \left(\bmod\ 2^{l(n)}\right).$$

Theorem 2. *With respect to probabilistic algorithms, and except for a polynomial factor in computation time, the modular XTSP is as hard as the XTSP itself.*

Sketch of proof: Let X be random, $B = \mathrm{Length}_A(X)$. A random Y with $\mathrm{Length}_A(Y) \equiv B$ modulo $2^{l(n)}$ satisfies the equation $\mathrm{Length}_A(Y) = B$ with probability $1/\Theta(n)$, since B is the sum of only n numbers, hence $B < 2^{l(n)} + n$. □

In the following we only regard the modular XTSP, thus all computations of the function Length_A are done modulo $2^{l(n)}$, where $l(n)$ is the number length of A.

3 One-Way Hash Functions

One-way hash functions are useful for electronic fingerprints. For $m > k(m)$ the functions $f_m : \{0,1\}^m \longrightarrow \{0,1\}^{k(m)}$ are *one-way hash functions*, if it is easy to compute $f_m(x)$ when given m and $x \in \{0,1\}^m$, but infeasible to find a $y \neq x$ in $\{0,1\}^m$ with $f_m(x) = f_m(y)$ for random x.

Theorem 3. *Let $l(n) < m = \lfloor \log_2((n-1)!) \rfloor$; if* Length_A *is a one-way function then* $f_m(X) = \text{Length}_A(X)$ *is a one-way hash function.*

Sketch of proof: If, given a random X, we can efficiently find some $Y \neq X$ with $\text{Length}_A(Y) = \text{Length}_A(X)$, then we can invert the function "Length_A". Let a target number B be given, then

- we chose a random Hamiltonian cycle X and let $B' = \text{Length}_A(X)$,
- we chose a random edge $i^* \rightarrow j^*$ of X and compute the matrix $A' = (a'_{i,j})$ with $a'_{i,j} = a_{i,j}$ for $(i \neq i^*$ or $j \neq j^*)$, $a'_{i^*,j^*} = a_{i^*,j^*} + B - B'$, consequently $\text{Length}_{A'}(X) = B$,
- and finally we compute a $Y \neq X$ with $\text{Length}_{A'}(X) = \text{Length}_{A'}(Y)$.

With nonnegligible probability the edge $i^* \rightarrow j^*$ is no part of the Hamiltonian cycle Y. In this the case $\text{Length}_A(Y) = B$. □

Universal one-way hash function were defined by Naor and Yung [9], who also outlined their application to digital signatures and fingerprints. For $m > k(m)$ the collections F_m of functions $f_{m,i} : \{0,1\}^m \longrightarrow \{0,1\}^{k(m)}$ constitute *families of universal one-way hash functions*, if given m, for any $x \in \{0,1\}^m$ and randomly chosen $f_{m,i} \in F_m$, it is easy to compute $f_{m,i}(x)$, but with overwhelming probability infeasible to find a $y \neq x$ in $\{0,1\}^m$ with $f_{m,i}(x) = f_{m,i}(y)$. Note that x is not random, but it's choice does not depend on $f_{m,i}$.

The Length_A-functions are families of universal one-way hash functions. Let X^* be fixed, A a random matrix and X a random Hamiltonian cycle. A is a distance matrix of the complete directed graph G_n, and by renaming (i.e. permuting) the vertices of G_n we can compute the distance matrix A^*, such that X and X^* "are the same cycle". If we can compute a Y^* with $\text{Length}_{A^*}(X^*) = \text{Length}_{A^*}(Y^*)$, this directly leads us to a Y with $\text{Length}_A(X) = \text{Length}_A(Y)$.

4 A Pseudo-Random Generator (PRG)

A PRG uses a short, "really random" input to generate a longer, "randomly looking" bit string S. For a *cryptographic PRG* it must be infeasible to distinguish between S and a "really random" bit string S', where each bit is generated independently according to the uniform distribution. I.e. there must be no probabilistic polynomial time algorithm, to distinguish between S and S' with probability significantly greater than 0.5.

Cryptographic PRGs are highly useful for many cryptographic applications. It is straightforward to use them as (secret key) stream ciphers.

Theorem 4. *Let $l(n) > 1 + log_2((n-1)!)$; if* Length_A *is a one-way function, then* $g(X) = \text{Length}_A(X)$ *is a cryptographic PRG.*

Sketch of proof: Assume there exists a polynomial time algorithm D' to distinguish between S and S' with probability $\frac{1}{2} + \frac{1}{p(n)}$, $p(n)$ a polynomial in n.

Then there also exists a polynomial time algorithm D to distinguish with overwhelming probability. We will use D to find out for any B, if there is an X with $\text{Length}_A(X) = B$.

Since $l(n) > 1 + \log_2((n-1)!)$, x is unique with nonnegligible probability. For any i, j, $i \neq j$ we randomly change $a_{i,j}$. As in the proof of theorem 3 we get a new distance matrix A'. We have $\text{Length}_{A'}(X) = B$ if and only if the edge $i \to j$ is no part of the Hamiltonian cycle X. In the other case with significant probability there is no Y with $\text{Length}_{A'}(Y) = B$. Thus we can use D to find the edges of X. □

5 On the Choice of A

Before one can apply our schemes, the coefficients $a_{i,j}$ of the matrix A must be fixed. We consider two natural ways to do this:

1. Generate n random points in a finite plane (or some higher dimensional space) and compute $a_{i,j}$ as the (e.g. Euclidean) distance between the points i and j. In order to save space one might store the coordinates of the points and compute the distances on demand.
2. Generate the $a_{i,j}$ as independent random numbers from $\{0, 2^{l(n)} - 1\}$, according to the uniform distribution.

Note that the first option leads to an undirected graph, i.e. $a_{i,j} = a_{j,i}$. Though we could cope with this, the first option is not recommendable. It is well known (cf. [3], section 37.2), that, if the triangle inequality holds for a TSP, there is a good deterministic approximation algorithm. But no such algorithm can exist for general TSPs, if NP≠P. There is no obvious way to make use of the triangle inequality for solving the XTSP. Nevertheless such inherent structures in the matrix A should be considered as possible weaknesses.

The second option forces us to generate and store a large number of random bits. A very convincing way to solve the generation problem is to use the first $l(n)n^2$ bits of the binary representation of $\pi - 3$. Other mathematical constants would do as well, if the resulting bits appear to be uniformly distributed.

6 How Infeasible is the XTSP?

As outlined in the introduction, we should not trust in the infeasibility of any problem simply because it is NP-complete. The "classical" TSP minimization problem is feasible for dimensions $n =$"several thousand", see Padberg and Rinaldi [10]. This is alarming!

On the other hand there are some reasons to believe that algorithms like Padberg's and Rinaldi's—which seams to be typical for all approaches to solve large-scale TSPs—are of few help for attacks against our cryptographic schemes:

1. The triangle inequality does hold for all solved problems.

2. The number representation for distances is limited. Padberg and Rinaldi used at most 64 bits; these were floating point numbers.
3. The results are achieved by branch–and–cut or branch–and–bound algorithms. The basic branch–and–... principle can roughly be described as follows:
   - Let a solution space S be given.
   - Divide S into subsets $S_1, S_2, \ldots, S_k$.
   - Compute lower and/or upper bounds for all solutions in S_i.
   - For all S_i do:
     - If the lower bound is too large (or the upper bound too low) then discard S_i
     - else apply branch–and–... on the solution space S_i.

   Clearly this is efficient if we can discard many subsets S_i at a high level of the recursion tree. In our case the target number B is computed as the length of some random Hamiltonian cycle X. Hence we can expect a nonnegligible fraction of all Hamiltonian cycles to be shorter than X and a nonnegligible fraction to be longer. Thus nearly all large subsets S_i of the solution space will contain both longer and shorter Hamiltonian cycles and we can discard almost none.

   So the XTSP appears to be a variant of the TSP, where branch–and–... works exceptionally bad.

Nevertheless some further research is necessary before we can suggest "probably secure" values for the parameters n and $l(n)$. The size of these parameters is, of course, essential for the speed of our schemes and for the size of the required memory.

Much more research is necessary before we can recommend our schemes for practical use. Any effort in attacking the schemes is appreciated!

References

1. T. Baritaud, M. Campana, P. Chauvaud, H. Gilbert, *On the security of the Permuted Kernel Identification Scheme*, in: Proc. Crypto '92, Springer LNCS 760, 305-311.
2. P. Camion, J. Patarin, *The Knapsack Hash Function proposed at Crypto '89 can be broken*, in: Proc. EuroCrypt '91, Springer LNCS 547, 39-53.
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1990.
4. I. Damgård, *Design Principles for Hash Functions*, in: Proc. Crypto '89, Springer LNCS 435, 416-427.
5. J. Georgiades, *Some Remarks on the Security of the Identification Scheme Based on Permuted Kernels*, in: J. Cryptology (1992), Vol. 5, 133-137.
6. R. Impagliazzo, M. Naor, *Efficient Cryptographic Schemes Provably as Secure as Subset Sum*, in: Proc. FOCS '89, 236-241.
7. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, D. B. Shmoys (eds.), *The Traveling Salesman Problem*, Wiley, 1985.

8. R. C. Merkle, M. Hellman, *Hiding information and Signature in Trapdoor Knapsack*, in: IEEE Trans. on Inf. Theory 24 (1978), 525-530.
9. M. Naor, M. Yung, *Universal One Way Hash Functions and Their Cryptographic Applications*, in: Proc. STOC '89, 33-43.
10. M. Padberg, G. Rinaldi, *A Branch–and–Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems*, in: Siam Review 33, No. 1 (1991), 60-100.
11. J. Patarin, P. Chauvaud *Improved Algorithms for the Permuted Kernel Problem*, in: Proc. Crypto '93, Springer LNCS 773, 391-402.
12. C. P. Schnorr, M. Euchner, *Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems*, in: Proc. FCT '91, 68-85.
13. A. Shamir, *An Identification Scheme based on Permuted Kernels*, in: Proc. Crypto '89, Springer LNCS 435, 606-609.
14. J. Stern, *A new identification scheme based on syndrome decoding*, in: Proc. Crypto '93, Springer LNCS 773, 13-20.
15. J. Stern, *Designing Identification Schemes with Keys of Short Size*. in: Proc. Crypto '94, Springer LNCS 839, 164-173.

This paper was published at:
Fast Software Encryption (1994),
Springer LNCS 1008, 298–304.