

# Polynomially Uncomputable Number-Theoretic Problems in Cryptography and Network Security

Song Y Yan & Glyn James

School of Mathematical and Information Sciences, Coventry University, Coventry CV1 5FB, UK  
s.yan@coventry.ac.uk

Gongyi Wu

College of Technical Information Science, Nankai University, Tianjin 300071, China

## Abstract

*By far the most important automated tool for network and communications security is encryption, but encryption itself needs security. As we know, many cryptosystems and cryptographic protocols are based their security on some computationally intractable mathematical problems. In this paper, we shall discuss some of the number-theoretic problems that are currently polynomially uncomputable and thus provide a good base for cryptography and network security.*

## 1 Introduction

Number Theory is one of the oldest and purest subject of mathematics. With the advent of modern computers, it is also very applicable, which may surprise Hardy [4] who regarded number theory as a useless subject of mathematics. One of the important features in the theory of numbers is that it abounds with many difficult problems which looks easy. For example, it is evident that every positive integer greater than 4 is the sum of two primes such as  $8 = 3+5$ , but no-one has ever been able to produce a proof; this is the infamous Goldbach conjecture. In public-key cryptography, however, we are interested in those number theoretic problems that apparently are solvable in theory but unsolvable in practice (i.e., in deterministic polynomial time). This class of problems are normally denoted by  $\mathcal{P}$ . On the other hand, the class of non-deterministic polynomial time problems are denoted by  $\mathcal{NP}$ . At present, we do not know whether or not  $\mathcal{P} = \mathcal{NP}$  (this is one of the seven Millennium Prize Problems, each with one million US dollars offered by the Clay Mathematical Institute in Boston) but conjectured  $\mathcal{P} \subset \mathcal{NP} = \mathcal{EXP}$ .

What we know at present is that problems in  $\mathcal{NP}$  can only be solved in exponential time, but this view may be changed in the future. It should be noted that in cryptography, we normally only use those problems that are *believed* but *have not been proved* to be computationally intractable. Anything beyond  $\mathcal{P}$  is computationally intractable. In terms of bit operations in number-theoretic computation (suppose  $N$  is the number to be computed,  $k$  a positive integer and  $c$  a small real constant), we have:

- Polynomial-time:

$$\mathcal{O}((\log N)^k) \longrightarrow \text{easy}$$

- Superpolynomial:

$$\mathcal{O}((\log N)^{c \log N}) \longrightarrow \text{rather hard}$$

- Subexponential:

$$\mathcal{O}\left(\left(c \sqrt[3]{\log N} \sqrt[3]{(\log \log N)^2}\right)\right) \longrightarrow \text{hard}$$

- Exponential:

$$\mathcal{O}(N^c) \longrightarrow \text{very hard.}$$

In this paper, we shall discuss some of those problems that are *believed* but *not proved* to be computationally intractable.

## 2 Factoring Related Problems

In this section we introduce several integer factorization related polynomially unsolvable problems in number theory and their applications in modern public-key cryptography.

## 2.1 The Integer Factorization Problem

Suppose we are giving two prime numbers, then it is easy to find the product of the two primes, no matter how big of these two primes:

$$p \cdot q \rightarrow n$$

On the other hand, if  $n$  is given, then it is very difficult to find its two prime factors:

$$n \rightarrow (p, q)$$

if  $n$  is large. That is,

$$\{n = p \cdot q\} \xrightarrow{\text{hard}} \{p, q\}$$

The RSA Security Inc has offered a prize of \$20,000 for the number RSA-193 (193 digits, 640 bits), the currently smallest unfactored RSA number on the list of the RSA challenge numbers:

31074182404900437213507500358885679300373460228\_42727545720161948823206440518081504556346829671\_72328678243791627283803341547107310850191954852\_90073377248227835257423864540146917366024776523\_46609

The fastest factoring algorithm at present is the Number Field Sieve (NFS) [6], runs in subexponential time  $\mathcal{O}(\exp(c(\log n)^{1/3}(\log \log n)^{2/3}))$ . The fundamental idea of the NFS is as follows. If we can find two integers  $x$  and  $y$  such that

$$x^2 \equiv y^2 \pmod{n}, \quad x \not\equiv \pm y \pmod{n}$$

then we find two nontrivial factors of  $n$  since  $1 < \gcd(n, x \pm y) < n$ . So in NFS, we first choose a monic polynomial  $f$  (irreducible over  $\mathbb{Z}$ ), a positive integer  $m$  with  $f(m) \equiv 0 \pmod{n}$ , an algebraic number  $\alpha \in \mathbb{C}$  (which is a root of  $f$ ), and a homomorphism  $\phi : \mathbb{Z}[\alpha] \mapsto \mathbb{Z}/n\mathbb{Z}$  via  $\phi(\alpha) = m$  such that for any  $g(x) \in \mathbb{Z}[x]$ ,  $\phi(g(\alpha)) \equiv g(m) \pmod{n}$ . We then try to find a set  $\mathcal{S}$  of polynomials  $g$  over  $\mathbb{Z}$  such that  $\prod_{g \in \mathcal{S}} g(\alpha) = \beta^2 \in \mathbb{Z}[\alpha]$  and  $\prod_{g \in \mathcal{S}} g(m) = y^2 \in \mathbb{Z}$ . Putting  $\phi(\beta) \equiv x \pmod{n}$ , we get

$$x^2 \equiv \phi(\beta)^2 \equiv \phi(\beta^2) \equiv \phi\left(\prod_{g \in \mathcal{S}} g(\alpha)\right) \equiv \prod_{g \in \mathcal{S}} g(m) \equiv y^2$$

all modulo  $n$ . Note that there is a quantum algorithm [7] that can factor a large number in polynomial-time  $\mathcal{O}(\log n)^{3+\epsilon}$ , but at present we do not have a quantum computer to run the quantum algorithm; a practical quantum computer may never be able to build (the

current quantum experiment can only factor  $15 = 3 \times 5$  which is totally hopeless). Since IFP is hard to solve, many cryptographic systems (such as the RSA system) are based their security on the intractability of this problem.

## 2.2 The RSA Problem

RSA is the first practical public-key cryptosystem based on the intractability of IFP. RSA works as follows:

- **Encryption:**  $C \equiv M^e \pmod{n}$
- **Decryption:**  $M \equiv M^d \pmod{n}$

where  $(e, n)$  is the public-key for encryption,  $d$  the decryption exponent,  $n = pq$  with  $p, q$  prime,  $ed \equiv 1 \pmod{\phi(n)}$ . Clearly, if one can factor  $n$ , he can find  $d$  by computing  $d \equiv 1/e \pmod{(p-1)(q-1)}$ , and hence find  $M$ . But the RSA problem is

$$\{e, n, C \equiv M^e \pmod{n}\} \xrightarrow{\text{hard}} \{M\}$$

It is evident that the RSAP is no harder than IFP, since if one can factor  $n$ , one can find  $M$  easily. This is the whole idea of the RSA cryptosystem.

## 2.3 The Square Root Problem

The SQRTP problem in  $\mathbb{Z}_n$  is, given  $(y, n)$ , find an integer  $x$  such that

$$y \equiv x^2 \pmod{n}$$

This is equivalently to say that we want to find an  $x$  such that

$$x \equiv \sqrt{y} \pmod{n}.$$

That is,

$$\{n, y \equiv x^2 \pmod{n}\} \xrightarrow{\text{hard}} \{x \equiv \sqrt{y} \pmod{n}\}$$

For example,  $\sqrt{3} = 1.732050808$  and  $\sqrt{4} = 2$ , however,  $\sqrt{3} \pmod{101}$  does not exist and  $\sqrt{4} \pmod{101} = 99$  (since  $99^2 = 9801 \pmod{101} = 4$ ). If  $n$  is prime number, the square root of  $y$  modulo  $n$  is easy to find in time  $\mathcal{O}((\log n)^4)$  [1]. If  $n$  is composite but its prime factors are known, it is still easy to find by using the Chinese Remainder Theorem. However, when  $n$  is a large composite (such as the RSA-193) whose prime factors are unknown, we need to factor  $n$  first. Thus, finding a square root modulo  $n$  is as hard as factoring a large composite integer.

## 2.4 The $k^{\text{th}}$ Root Problem

The SQRTP can be easily extended to the  $k$ th Root Problem (KRTP): given  $(k, y, n)$ , find an integer  $x$  such that

$$y \equiv x^k \pmod{n}$$

This is equivalently to say that we want to find an  $x$  such that

$$x \equiv \sqrt[k]{y} \pmod{n}.$$

That is,

$$\{n, k, y \equiv x^k \pmod{n}\} \xrightarrow{\text{hard}} \{x \equiv \sqrt[k]{y} \pmod{n}\}$$

If  $n$  is a prime number or the prime factorization of  $n$  is known, the  $k$ th root problem is easy to solve [10]:

## 2.5 The Quadratic Residuosity Problem

An integer  $a$  is a quadratic residue modulo  $n$ , denoted by  $a \in Q_n$ , if  $\gcd(a, n) = 1$  and there exists a solution  $x$  to the congruence  $x^2 \equiv a \pmod{n}$  (note that we do not need to find the  $x$ , just to decide whether or not such an  $x$  exist), otherwise  $a$  is a quadratic non-residue modulo  $n$ , denoted by  $a \in \overline{Q}_n$ . The Quadratic Residuosity Problem may be stated as:

Given positive integers  $a$  and  $n$ , decide whether or not  $a \in Q_n$ .

if  $n$  is prime then

$$a \in Q_n \iff \left(\frac{a}{n}\right) = 1,$$

and if  $n$  is composite, then

$$a \in Q_n \implies \left(\frac{a}{n}\right) = 1,$$

but

$$a \in Q_n \not\Leftarrow \left(\frac{a}{n}\right) = 1,$$

however

$$a \in \overline{Q}_n \iff \left(\frac{a}{n}\right) = -1.$$

Let

$$J_n = \{a \in (\mathbb{Z}/n\mathbb{Z})^* : \left(\frac{a}{n}\right) = 1\},$$

then

$$\tilde{Q}_n = J_n - Q_n.$$

Thus,  $\tilde{Q}_n$  is the set of all pseudosquares modulo  $n$ ; it contains those elements of  $J_n$  that do not belong to  $Q_n$ . The Quadratic Residuosity Problem can then be further restricted to:

Given a composite  $n$  and an integer  $a \in J_n$ , decide whether or not  $a \in Q_n$ .

For example, when  $n = 21$ , we have  $J_{21} = \{1, 4, 5, 16, 17, 20\}$  and  $Q_{21} = \{1, 4, 16\}$ , thus  $\tilde{Q}_{21} = \{5, 17, 20\}$ . So, the QRP problem for  $n = 21$  is actually to distinguish squares  $\{1, 4, 16\}$  from pseudosquares  $\{5, 17, 20\}$ . The only method we know for distinguishing squares from pseudosquares is to factor  $n$ ; since integer factorization is computationally infeasible, the QRP problem is believed to be computationally infeasible. This is exactly the base of several cryptographic systems, including the Goldwasser and Micali probabilistic encryption [3].

## 2.6 The $k^{\text{th}}$ Power Residuosity Problem

The idea of quadratic residues can be extended to the  $k^{\text{th}}$  (higher) power residues. Let  $a, n$  and  $k$  be positive integers with  $k \geq 2$ . Suppose  $\gcd(a, n) = 1$ , then  $a$  is called a  $k$ th (higher) power residue of  $n$  if there is an  $x$  such that

$$x^k \equiv a \pmod{n}.$$

The set of all  $k$ th (higher) power residues is denoted by  $K(k)_n$ . If the congruence has no solution, then  $a$  is called a  $k$ th (higher) power nonresidue of  $n$ . The set of such  $a$  is denoted by  $\overline{K}(k)_n$ . For example,  $K(9)_{126}$  would denote the set of the 9th power residues of 126, whereas  $\overline{K}(5)_{31}$  the set of the 5th power nonresidue of 31. Thus, we have the following analog definition of quadratic residuosity problem (QRP) for the  $k$ th (higher) power residuosity problem (KRP): given integer  $a$  and  $m$ , decide whether or not  $a \in K_m$ . The Legendre and Jacobi symbols for quadratic residues can be extended to the  $k$ th power residues. So, cryptographic schemes can of course be based on the computational intractability of the  $k$ th power residuosity problem [9].

## 3 Logarithm Related Problems

Now we move on to the logarithm related problems and their applications in cryptography.

### 3.1 The Discrete Logarithm Problem

The DLP for multiplicative group  $\mathbb{Z}_p^*$  over the finite field  $\mathbb{Z}_p$  can be described as follows. Given  $(x, y, p)$ , find an integer  $k$  such that

$$y \equiv x^k \pmod{p}$$

That is,

$$\{n, x, y \equiv x^k \pmod{n}\} \xrightarrow{\text{hard}} \{k\}$$

The fastest method in use for discrete logarithms is the index calculus, which is a wide range of methods, including the NFS for IFP. So, essentially any method that can be used for IFP can be used for DLP by some modifications and the complexity of method for both problems will be almost the same. As the complexity of any method in index calculus is sub-exponential, the DLP is only polynomially unsolvable, and hence can be used as a base for cryptography. In fact, the first public-key idea of Diffie-Hellman-Merkle and the famous US government's digital signature algorithm/standard (DSA/DSS) are all based their security on the intractability of DLP.

### 3.2 The Diffie-Hellman-Merkle Problem

The DHM key-change system works in the following way:

- 1) A prime  $q$  and a generator  $g$  are made public (assume all users have agreed upon a finite group over a fixed finite field  $\mathbb{F}_q$ ),
- 2) Alice chooses a random number  $a \in \{1, 2, \dots, q-1\}$  and sends  $g^a \pmod q$  to Bob,
- 3) Bob chooses a random number  $b \in \{1, 2, \dots, q-1\}$  and sends  $g^b \pmod q$  to Alice,
- 4) Alice and Bob both compute  $g^{ab} \pmod q$  and use this as a private key for future communications.

Clearly, an eavesdropper has  $g, q, g^a \pmod q$  and  $g^b \pmod q$ , so if he can take discrete logarithms, he can calculate  $g^{ab} \pmod q$  and comprise communications. However, the DHM problem is

$$\{g, q, g^a \pmod q, g^b \pmod q\} \xrightarrow{\text{hard}} \{g^{ab} \pmod q\}$$

because

$$\{g, q, g^a \pmod q\} \xrightarrow{\text{hard}} \{a\}$$

or

$$\{g, q, g^b \pmod q\} \xrightarrow{\text{hard}} \{b\}$$

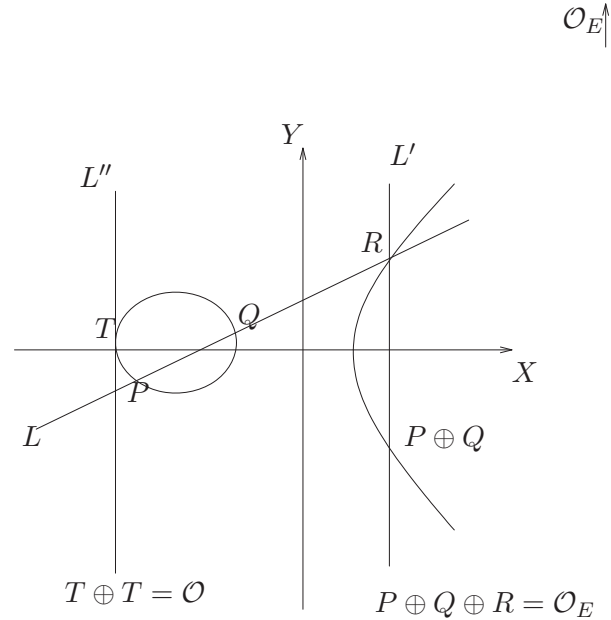
This is again the whole idea of DHM key-exchange scheme.

### 3.3 The Elliptic Curve Discrete Logarithm Problem

The DLP can be naturally extended to the elliptic curves over a finite field, for which, we call it the Elliptic Curve Discrete Logarithm Problem (ECDLP). An elliptic curve is a plan algebraic curve given by

$$E : y^2 = x^3 + ax + b,$$

If we can find two points on the curve such as  $P$  and  $Q$ , we can always find a third  $P \oplus Q$  (see Figure 1). There



is a very convenient algebraic formula for calculating the third point  $P \oplus Q$ . Let  $P_1 = (x_1, y_1)$ ,  $P_2 = (x_2, y_2)$  be points on the elliptic curve defined above, then  $P_3 = (x_3, y_3) = P_1 \oplus P_2$  on  $E$  may be computed by

$$P_1 \oplus P_2 = \begin{cases} \mathcal{O}_E, & \text{if } x_1 = x_2 \text{ \& } y_1 = -y_2 \\ (x_3, y_3), & \text{otherwise.} \end{cases}$$

where

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1)$$

and

$$\lambda = \begin{cases} \frac{3x_1^2 + a}{2y_1}, & \text{if } P_1 = P_2, \\ \frac{y_2 - y_1}{x_2 - x_1}, & \text{otherwise.} \end{cases}$$

Now we notice that it should be easy to calculate

$$Q \equiv kP \pmod p$$

using a method similar to the fast exponentiation  $y \equiv x^k \pmod{n}$ . But the problem, however, is that given  $(P, Q, p)$ , it is very difficult to find the  $k$ . That is,

$$\{p, P, Q \equiv kP \pmod{n}\} \xrightarrow{\text{hard}} \{k\}.$$

As mentioned, although for DLP there is no polynomial-time algorithm, there are subexponential-time algorithms. For ECDLP, unfortunately, there is even no subexponential-time algorithm. It is exactly this feature that makes ECDLP a good security base for cryptographic systems. Recently, Silverman [8] developed an algorithm, called xedni calculus, for ECDLP, but it does not run in subexponential-time; readers are suggested to consult [10] for more information.

## 4 Conclusion

In this paper, we discussed several polynomially unsolvable number-theoretic problems and their significance in cryptography and network security. As can be seen, almost all the problems such as SQRT, kRP, QRP and kPRP are related to IFP, so, if IFP can be solved in polynomial time, many other related problems can also be solved in polynomial time. We noted also that ECDLP is rather different from DLP and IFP, but there is an algorithm, Xedni calculus, that can be used to IFP, DLP and ECDLP, although it is inefficient. It may turn out to be that someday someone may develop a practical algorithm than Xedni calculus that can be used for IFP, DLP, ECDLP. In that case, all the practically useful cryptographic schemes such as RSA, DSA and ECDSA will be no more secure. It is urgent to find some other alternative mathematical problems for IFP, SQRT, QRP, DLP and ECDLP.

## References

- [1] H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Springer-Verlag, 1993.
- [2] M. R. Garey and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.
- [3] S. Goldwasser and S. Micali, “Probabilistic Encryption”, *Journal of Computer and System Sciences*, **28** (1984), 270–299.
- [4] G. H. Hardy, *A Mathematician’s Apology*, Cambridge University Press, 1940 (1979 Printing).
- [5] N. Koblitz, “A Survey of Number Theory and Cryptography”, in: *Number Theory*, Edited by P. Bambah, V. C. Dumir and R. J. Hans-Gill, Birkhäuser, 2000, 217–239.
- [6] A. K. Lenstra and H. W. Lenstra, Jr., (editors), *The Development of the Number Field Sieve*, Lecture Notes in Mathematics **1554**, Springer-Verlag, 1993.
- [7] P. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”, *SIAM Journal on Computing*, **26**, 5(1997), 1484–1509.
- [8] J. H. Silverman, “The Xedni Calculus and the Elliptic Curve Discrete Logarithm Problem”, *Designs, Codes and Cryptography*, **20**, 2000, 5-40.
- [9] S. Y. Yan, “A New Cryptographic Scheme based on the kth Power Residuosity Problem”, 15th British Colloquium for Theoretical Computer Science (BCTCS15), Keele University, 14-16 April 1999.
- [10] S. Y. Yan, *Number Theory for Computing*, 2nd Edition, Springer-Verlag, 2002.