Send Message into a Definite Future

Wenbo Mao
Trusted E-Services
HP Laboratories Bristol
HPL-1999-135
27th October, 1999*

time-lock puzzle, timed-release cryptography, zero-knowledge proof

Rivest et al proposed a time-lock puzzle scheme for encrypting messages which can only be decrypted in the future. Such a puzzle specifies an algorithm for decrypting the message locked in and the specified algorithm has a well understood time complexity. However, that time-lock puzzle scheme does not provide a means for one to examine whether a puzzle has been formed in good order. Consequently, one may foolishly waste a lengthy time on trying to solve an intractable problem. This weakness prohibits that scheme from applications that involve mutually untrusted parties. We propose a new time-lock puzzle scheme which includes an efficient protocol that allows examination of the time needed for decrypting the message locked in.

*Internal Accession Date Only

Send Message into a Definite Future

Wenbo Mao

Hewlett-Packard Laboratories,
Filton Road, Stoke Gifford, Bristol BS34 8QZ, United Kingdom
wm@hplb.hpl.hp.com

Abstract. Rivest et al proposed a time-lock puzzle scheme for encrypting messages which can only be decrypted in the future. Such a puzzle specifies an algorithm for decrypting the message locked in and the specified algorithm has a well understood time complexity. However, that time-lock puzzle scheme does not provide a means for one to examine whether a puzzle has been formed in good order. Consequently, one may foolishly waste a lengthy time on trying to solve an intractable problem. This weakness prohibits that scheme from applications that involve mutually untrusted parties. We propose a new time-lock puzzle scheme which includes an efficient protocol that allows examination of the time needed for decrypting the message locked in.

1 Introduction

Rivest et al [5] proposed a timed-release crypto puzzle scheme (time-lock puzzle). The goal is to encrypt a message so that it cannot be decrypted until a pre-determined period of time has passed.

Several applications have been observed in [5]:

- A bidder in an auction wants to seal his bid so that it can only be opened after the bidding period is closed.
- A homeowner wants to give his mortgage holder a series of encrypted mortgage payments. These might be encrypted digital cash with different decryption dates, so that one payment becomes decryptable (and thus usable by the bank) at the beginning of each successive month.
- A key-escrow scheme can be based on timed-release crypto, so that the government can get the message keys, but only after a fixed, pre-determined period.
- An individual wants to encrypt his diaries so that they are only decryptable after fifty years (when the individual may have forgot the decryption key).

The time-lock puzzle scheme proposed in [5] has its security based on the computational problem of finding an element in a multiplicative group modulo a composite integer. The scheme provides a prescribed instruction leading to the target element for solving the puzzle (the element will reveal a key for decrypting the locked message). If a puzzle solver follows the instruction provided, the problem has a well-understood time complexity. Otherwise, the problem is believed to be intractable if the composite modulus used is sufficiently large.

In this paper we shall point out a major weakness of that time-lock puzzle scheme: the time required to solve a puzzle is not pre-determinable from a solver's point of view. In fact, apart from the puzzle maker, no one can be sure if a puzzle is indeed solvable until it has actually been solved. A time-lock puzzle in usual applications typically requires a lengthy time to solve. So to start solving a puzzle without knowing if it has been created in good order can mean to foolishly risk wasting a lengthy period of time, unless the puzzle maker is trusted to have created the puzzle as bona-fide. However, in most applications where a timed-release crypto puzzle finds a role to play, a puzzle maker should not be trusted by a solver (this is the case in the first three applications listed above). Therefore, a time-lock puzzle scheme which does not allow a solver to pre-determine solveability within the time length declared has a serious limitation in applications.

We will propose a new time-lock puzzle scheme which can allows a solver to examine with confidence the time length that is needed to solve a puzzle.

2 The previous time-lock puzzle

We first introduce the time-lock puzzle scheme proposed in [5], and then analyse a weakness with it.

Suppose Alice has a message M that she wants to encrypt with a time-lock puzzle for T seconds. She generates a composite modulus $n = pq$ with p and q being two large primes. She sets t such that $t = TS$ with S being the number of squarings modulo n per second that can be performed by the solver. Alice then generates a random key K for a conventional cryptosystem. She encrypts M with the key K to form ciphertext C_M. She now picks a random element a modulo n, and encrypts K as

$$C_K = K + a^e (\bmod n),$$

where

$$e = 2^t (\bmod \phi(n)).$$

She finally outputs the time-lock puzzle (n, a, t, C_K, C_M) and erases any other variables created during this computation. With the knowledge of $\phi(n)$, Alice can construct the puzzle efficiently.

Assume that finding M from C_M without the correct key is computationally infeasible, so is computing $\phi(n)$ from n. Then computing $a^{2^t} (\bmod n)$ as recommended by the scheme will be the only known way to solve the puzzle. Once $a^{2^t} (\bmod n)$ is reached, K will be revealed which will further lead to decryption of C_M. This computation requires t steps of squaring modulo n. Obviously, t must be a tractable quantity.[1]

[1] The idea of repeated squaring an element in a group modulo a composite integer has also been used in a key escrow scheme with a time control feature [1], which utilises

An advantage of this puzzle scheme is that the process of repeated squaring is "intrinsically sequential". There seems no obvious way to parallelise it to any large degree. So the time needed to solve a puzzle can be well controlled by Alice.

We should note that in this scheme there is no control whatsoever over Alice in terms of enforcing her to form a puzzle to be bona fide. She can introduce errors into a puzzle which is undetectable to a solver (let Bob be the solver) before he realises that he has been spending a lengthy period of time with fruitless result. For instance, Alice can set t in the puzzle to be infeasibly large. Note that it is trivially easy for Alice to set the order of a modulo n and that of 2 modulo $\phi(n)$ to be sufficiently large; this will guarantee that t can also be sufficiently large without making e and a^e to show a sign of cycle. Thus to reach $a^e (\bmod n)$ by following Alice's instruction will take infeasibly many steps.

In usual applications of timed-release cryptography (e.g., key escrow with time-delayed key recovery feature), the puzzle maker's successful cheating means a total defeat of the system security. The problem here is not that Alice is untrustworthy; it is the absence of a means for Alice to show her honesty. Thus, in applications where the puzzle maker and solver do not trust each other (this is likely, for instance, these two parties should not trust each other in the first three applications listed in Section 1), [5] suggests that a commonly trusted third party be used to check that Alice has formed a puzzle correctly. The use of a commonly trusted third party greatly limits the usefulness of the time-lock puzzle scheme.

3 A new time-lock puzzle scheme

The new time-lock puzzle scheme proposed here is equipped with a protocol that allows Alice to efficiently prove her correctness in creation of a puzzle. Solution of a puzzle now means successful factoring of a large composite integer that she has generated for the puzzle. So the new scheme is in fact a timed-release factorisation of a large integer. We relate the factorisation problem to that of computing a "small" discrete logarithm which has a well understood and controllable time complexity.

To create a puzzle, Alice shall setup $n = pq$ with p and q being large primes such that -1 has the positive Jacobi symbol modulo n. (The Jacobi symbol of an element modulo n can be efficiently evaluated.) Let $\left(\frac{x}{n}\right)$ denote the Jacobi symbol of x modulo n. Then we require

$$\left(\frac{-1}{n}\right) = 1. \tag{1}$$

Alice shall disclose an element of a hidden order modulo n. (Almost all elements in the multiplicative group modulo n have hidden orders.) Let e be this element and t be the hidden order. This means we require

$$e^t \equiv 1 \,(\bmod n). \tag{2}$$

the difficulty of computing a square-root of a groups element as a means to prevent unauthorised wire-tapping of communications taken place prior to a warranted date.

In addition, we require that the element e be chosen to satisfy

$$\left(\frac{e}{n}\right) = -1. \tag{3}$$

Since Jacobi symbol is multiplicative, we have

$$1 = \left(\frac{1}{n}\right) = \left(\frac{e^t \,(\mathrm{mod}\, n)}{n}\right) = \left(\frac{e}{n}\right)^t = (-1)^t.$$

Therefore the secret order t must be even.

The following method can be used to setup n, e, and t to satisfy the requirements in (1-3).

First, Alice chooses two primes u and v. She should then test the primality of the following two quantities

$$p = 2p'u + 1, \qquad q = 2q'v + 1, \tag{4}$$

for some odd numbers p' and q'. The procedure completes once both p and q are found to be primes. Let $n = pq$ and $t = 2uv$. Notice that u, v, p' and q' are odd; we have

$$\left(\frac{-1}{n}\right) = \left(\frac{-1}{p}\right)\left(\frac{-1}{q}\right) = (-1)^{(p-1)/2}(-1)^{(q-1)/2} = (-1)^{p'u}(-1)^{q'v} = 1.$$

So requirement (1) is met.

Alice then finds an element a modulo p of order $p-1 = 2p'u$, and an element b modulo q of order $(q-1)/2 = q'v$. It is easy for her to find such a and b. Let

$$c = a^{p'} \bmod p,$$

$$d = b^{2q'} \bmod q.$$

We know

$$c^{2u} \equiv 1 \,(\mathrm{mod}\, p),$$

$$d^{v} \equiv 1 \,(\mathrm{mod}\, q).$$

Applying the Chinese remainder theorem, Alice can compute $e < n$ satisfying

$$e \equiv c \,(\mathrm{mod}\, p) \quad \text{and} \quad e \equiv d \,(\mathrm{mod}\, q).$$

Obviously,

$$e^t \equiv e^{2uv} \equiv 1 \,(\mathrm{mod}\, n).$$

This meets the requirement (2).

The fact that a generates a group of $p-1$ elements with half of them being quadratic non-residues renders itself to be a quadratic non-residue (modulo p). Therefore

$$\left(\frac{a}{p}\right) = -1.$$

Further, since p' is odd, we know

$$\left(\frac{c}{p}\right) = \left(\frac{(a^{p'} \bmod p)}{p}\right) = \left(\frac{a}{p}\right)^{p'} = (-1)^{p'} = -1.$$

But d is a square number (modulo p), therefore

$$\left(\frac{d}{q}\right) = 1.$$

These yield

$$\left(\frac{e}{n}\right) = \left(\frac{e \bmod p}{p}\right)\left(\frac{e \bmod q}{q}\right) = \left(\frac{c}{p}\right)\left(\frac{d}{q}\right) = (-1)(1) = -1.$$

This meets the requirement (3).

Using e, n and t, Alice can prove her knowledge of t and at the same time showing the size of t without disclosing t to a verifier (Bob). This is via an interactive knowledge proof protocol (which is specified in A, and the protocol is due to Damgård [2]). The protocol is very efficient. There are also efficient protocols for Alice to show that n is the product of two primes (e.g., [6]).

Proof of the size of t and the two-prime-product structure of n are all what Alice has to do in order to show her honesty in construction of her time-lock puzzle. When t is relatively "small" ($\leq 2^{130}$), such a proof indicates the time needed for finding this secret order of e. The time complexity can be measured by $\sqrt{t}$ based on using the best known algorithms: Shank's baby-step-giant-step algorithm (e.g., page 105 [3]), Pollard's catching-kangaroo algorithm [4]. These algorithms make use of the fact that the discrete logarithms to be extracted are much smaller than the size of the (main) group in question.

After Bob has verified and accepted Alice's proofs, the time-lock puzzle will be accepted as the pair (e, n). Below we show that any message encrypted using n (e.g., using the RSA cryptosystem) can be decrypted after the puzzle is solved. The instruction to solve the puzzle is to extract t as the discrete logarithm of 1 to the base e modulo n.

Suppose that t has been revealed with $\sqrt{t}$ steps of computation. Then Bob can compute

$$f = e^{t/2} \bmod n.$$

Here $t/2 = uv$ is odd. We know from (2)

$$f^2 \equiv e^t \equiv 1 \pmod{n}.$$

We also know from (3)

$$\left(\frac{f}{n}\right) = \left(\frac{e^{uv} \pmod{n}}{n}\right) = \left(\frac{e}{n}\right)^{uv} = (-1)^{uv} = -1,$$

while from (1)

$$\left(\frac{1}{n}\right) = \left(\frac{-1}{n}\right) = 1.$$

Therefore

$$f \not\equiv \pm 1 \pmod{n}.$$

From this we can conclude that f must be a non-trivial square-root of 1. Since $f^2 \equiv 1 \pmod{n}$, we can write

$$(f+1)(f-1) = kn, \tag{5}$$

for some k. With $f \neq \pm 1$ and $0 < f < n$, we know

$$0 < f - 1 < f + 1 < n.$$

Thus (5) indicates that either $f-1$ or $f+1$ must contains a non-trivial factor of n.

So n is factored and the time-lock puzzle solved! The solving time is set under the control of an evidence which is thoroughly examinable by the solver before setting out to solve the puzzle.

4 Discussion

We discuss a number of issues which are related to the proposed scheme.

i) The time complexity measurement for extracting discrete logarithm (i.e., extracting t needs $T = \sqrt{t}$ steps of multiplications modulo n) reaches the lowest known to date on exploitation of extracting "small" discrete logarithms. Any algorithm using fewer operations will provide a breakthrough improvement on solving the "small" discrete logarithm problem.

ii) Since squaring modulo n takes exactly the same time measurement as multiplication modulo n, the time control T in our scheme (which is the number of multiplication) will be exactly equal to that in [5] which is the number of squaring modulo n.

iii) Space complexity (measured in number of bits need to be stored): Pollard's catching-kangaroo algorithm [4] is $O(\log t \log n)$ while Shank's baby-step-giant-step algorithm (e.g., page 105 [3]) is $O(t^{1/2} \log n)$. So for a large t (e.g., $t \approx 2^{100}$) the space requirement of the catching-kangaroo algorithm is trivial while that of the baby-step-giant-step algorithm becomes prohibitive for practical use of the algorithm. For a small t, the baby-step-giant-step algorithm is preferred because it is deterministic.

iv) Neither of the two algorithms that we suggest to use can be parallelised very well. Running m processes independently gives a speedup of only $\sqrt{m}$. So for instance, in order to achieve a speedup of one thousand folds by parallelisation, one million processes are needed. Van Oorschot and Wiener [7] suggested a parallelised catching-kangaroo algorithm which uses m kangaroos, each digs a hole after many jumps (rather than after each jump as in Pollard's original algorithm). That algorithm achieves a linear speedup; however, its space complexity becomes $O(t^{1/2} \log n)$ (number of bits to be stored in a central storage shared by all processors). When t is large (i.e., when parallelisation is most effective), this space requirement is prohibitive for a practical use of the parallelised algorithm.

v) Pollard's rho algorithm for extracting discrete logarithms (e.g., page 106 [3]) is not usable here since it requires to know the order of the group (the discrete logarithm of 1 to the base e) in first place.

vi) For $t \leq 2^{130}$, $T \leq 2^{65}$ which allows the scheme to be usable in applications which fall in a wide range of time frame. However, for a very long term time control, one should consider the effect of Moore's Law. This consideration should be universally observed for any time-lock puzzle which is based on a computational complexity problem, as long as Moore's Law remains to be true.

vii) Alice can set the size of n to be arbitrarily large by setting large p' and q' in (4). They can also be selected as primes, which makes $p-1$ and $q-1$ to be non-smooth, a standard means for countering a number of known factorisation algorithms based on exploiting the smoothness of $\phi(n)$. Under such a setting, to factor n without using the prescribed instruction is well understood to be more costly.

5 Conclusion

We have constructed an integer factorisation-based time-lock puzzle scheme. The construction allows the puzzle maker to prove to a solver the time needed for solving a puzzle.

Acknowledgments

I would like to thank my colleague Nigel Smart for discussions which led to the study of the parallelised catching-kangaroo algorithm.

References

1. Burmester, M., Desmedt, Y. and Seberry, J. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically). Extended abstract. Advances in Cryptology — Proceedings of ASIACRYPT 98 (K. Ohta and D. Pei eds.), Lecture Notes in Computer Science, Springer-Verlag 1514 (1998) pages 380–391.
2. Damgård, I.B. Practical and provably secure release of a secret and exchange of signatures. Advances in Cryptology: Proceedings of EUROCRYPT 93 (T. Helleseth, ed.), Lecture Notes in Computer Science, Springer-Verlag, 765 (1994) pages 201–217.
3. Menezes, A.J., van Oorschot, P.C. and Vanstone, S.A. *Handbook of Applied Cryptography.* CRC Press. 1997.
4. Pollard, J.M. Monte Carlo method for index computation (mod p), *Mth. Comp.*, Vol.32, No.143 (1978), pages 918–924.
5. Rivest, R.L., Shamir, A. and Wagner, D.A. Time-lock puzzles and timed-release crypto. Manuscript. Available at (`http://theory.lcs.mit.edu/~rivest/RivestShamirWagner-timelock.ps`).

6. van de Graaf, J. and Peralta, R. A simple and secure way to show the validity of your public key. Advances in Cryptology — Proceedings of CRYPTO 87 (E. Pomerance, ed.), Lecture Notes in Computer Science, Springer-Verlag 293 (1988) pages 128–134.
7. van Oorschot, P.C. and M.J.Wiener M.J. Parallel collision search with cryptanalytic applications. *J. of Cryptology*, Vol.12, No.1 (1999), pages 1–28. (http://theory.lcs.mit.edu/ rivest/RivestShamirWagner-timelock.ps).

A Proof of the size of a secret order

The basic technique is due to Damgård [2]. We specify a simplified variation to suit our case of showing the order of an element. In our variation, the number in question is protected in computational (statistical) zero-knowledge.

Let t be in the interval $[a, b] = \{x | a \leq x \leq b\}$ where $c = b - a$ and $c \leq a$. In protocol Size specified below, Alice can convince Bob that t, the discrete logarithm of 1 modulo n to the base e, is in the interval $[a - c, b + c]$.
Protocol Size(e, n)
Execute the following k times:

1. Alice picks $0 < s_1 < c$ at uniformly random, and sets $s_2 = s_1 - c$;
she sends to Bob the pair
$$E_1 = e^{s_1} \bmod n, \quad E_2 = e^{s_2} \bmod n;$$
2. Bob selects $d = 0$ or $d = 1$ at uniformly random, and sends d to Alice;
3. Alice sets
$$\begin{array}{ll} u_1 = s_1, \quad u_2 = s_2 & \text{for } d = 0; \\ u_1 = s_1 + t, u_2 = s_2 + t & \text{for } d = 1; \end{array}$$
and sends u_1, u_2 to Bob;
4. Bob verifies
$$E_1 \equiv e^{u_1} (\bmod n), \quad E_2 \equiv e^{u_2} (\bmod n),$$
and
$$\begin{array}{ll} 0 < u_1 < c, \quad -c < u_2 < 0 & \text{for } d = 0; \\ c \leq u_1 \leq b + c, 0 \leq u_2 \leq b + c & \text{for } d = 1; \end{array}$$

In Bob's evaluation of $e^x \bmod n$, the case of $x < 0$ should be evaluated by performing $(1/e)^{|x|} \bmod n$.

Set, for instance, $[a, b] = [2^{\ell-1}, 2^\ell]$. Then $c = 2^{\ell-1}$. Size will prove that the order of the element e has a size not exceeding $\ell + 1$ binary bits. The probability for this to hold is at least $1 - 1/2^k$.