Time-Lock Puzzle with Examinable Evidence of Unlocking Time
(Transcript of Discussion)

Wenbo Mao

Good afternoon. I’ll be sticking to the auditability theme. This is about a protocol which was proposed by Rivest, Shamir and Wagner. It’s a timelock puzzle and to start with I will look at what is a timelock puzzle and what is its use, and then look at the RSW scheme, and then it will be an obvious requirement for auditability, to establish that the puzzle can really be solved within the stated time.

Before going into details I look at what the puzzle actually is. It is timed-release cryptography, which takes a very long time, or any specifiable length of time, to solve. Once it’s solved you then know some bits of crypto. It is based on RSA, and now there is argument about how to name RSA, somebody says it’s the alleged trademark of the cryptography used, so somebody else says it’s secret order, like discrete logarithm, address the problem rather than the inventor’s name.

Now the applications of time-release cryptography. Obviously there are several, say a bidder wants to seal a bid for a bidding period, another thing is sending messages to the future, a secret to be read in 50 years’ time, and another thing is key escrow architecture. Key escrow is this thing where there is a requirement to escrow some keys so that they can be recovered, and the danger is vast scale intrusion. So with timed-release cryptography it will take some time to produce a key, although we mustn’t waste a tremendous amount of time, but vast scale penetration becomes infeasible, becomes an individual criminal does not have the resources, so this is an example of a real application.

Now look at the RSW scheme. It is based on a secret order to an element. Suppose Alice has a secret to encrypt with a timelock puzzle for \( t \) units of time to solve. She generates two big primes \( p, q \) and multiplies them to obtain \( n \), and then picks a random session key \( K \) and encrypts with this the message \( M \) using conventional key cryptography to get \( C_M \). Then she encrypts the session key \( K \) using RSA, by adding \( a^e \) modulo \( n \) to give \( C_K \). Here \( a \) is a random element and this exponent \( e \) is defined as \( 2^t \mod \phi(n) \) where \( t \) is the number of timesteps needed to solve the puzzle. Since Alice generated \( p \) and \( q \) she can compute this \( e \) easily, whereas without knowing the factorization you cannot compute \( \phi(n) \). Now \( C_M \) and \( a \) and \( C_K \) are published, so this triple becomes the timelock puzzle. So if we analyse it we know that to decrypt message \( M \) from \( C_M \) you need obviously the correct key, assume this, and to decrypt \( K \) from \( C_K \) you need to compute \( a^e \) mod \( n \). Without knowing the factorization of \( n \) it seems that the only known way to compute \( a^e \) is by a repeated squaring of \( a \), so that is \( t \) multiplications.
This procedure is intrinsically sequential, since there is no way to parallelise it. No matter how you compute, you may have any tool, it will actually still get the same number of operations, so it’s sequential.

**Michael Roe:** There’s some fine-grained parallelism in computing a square, because if I write the number out in a power series and then two of them are multiplied together, it looks like I can do in parallel at least some of those multiplies. We’ve still got the additions which I can’t parallelise but I can do a lot of the multiplication, so I can gain a fair bit on squaring on a parallel machine. I think you’re right in the sense that there is a limiting part through the middle of it that can’t be parallelised but I can gain about $\log n$ by having parallel processes that work on different parts of it. Beyond that I don’t get any speed up at all, so as long as $t$ is much bigger than $\log n$ then, yes, intrinsically sequential.

**Bruce Christianson:** The argument is that the time is eventually linear in $t$.

**Matt Blaze:** It seems that it may be non parallelizable for one instance, but somebody who wants to parallelise a large number of these can certainly do a large number in parallel.

**Reply:** OK, but for one instance you cannot do it, so that’s not an attack to this model.

**Virgil Gligor:** So we are treating here the problem of time release, I want to release, for example a key, at a certain time, and to make sure the bits are not opened earlier than a certain date. Can I do that as a matter of the protocol as opposed to as a matter of encryption? There is a protocol which is presented at the annual Computer Security Applications Conference last December by Michiharu Kudo from IBM Tokyo Research Lab on a time release protocol, and this paper was purely based on classic public key cryptography, there was absolutely nothing new in there. So that gives the impression that you could do more or less what you do with this new sort of encryption scheme without this kind of encryption.

**Reply:** Oh, yes, I guess that’s true, this is only one way.

**Virgil Gligor:** So perhaps it would be good to think about the differences between solving the problem with conventional cryptography as a matter of a protocol, or solving it with a new form of crypto.

**Ross Anderson:** There’s a small point there, it’s perhaps not prudent to say that the time is linear in $t$. There’s a nice point that Serge Vaudenay made, that if you ask a really naive first year undergraduate how much longer it will take to search 128 bits of key compared with 64 bits of key, then you expect the student will say twice as long. Now given Moore’s law, this is actually correct because if you’re given $1M$ and 20 years to do key search, you do nothing at all for 19 years and six months and then you do the whole thing, and so your key search power is up exponentially and so it does take you twice as long to search 128 bits than 64. So you have to be mildly careful, I think, not to say