Lucre: Anonymous Electronic Tokens v1.3

Ben Laurie
ben@algroup.co.uk

July 23, 2000

1 Introduction

This is a revised version of the theory of blinded coins that may not violate Chaum's patent[1], based on the original work by David Wagner, and conversations with Ian Goldberg, David Molnar, Paul Barreto and various Anonymouses.

2 Coins

2.1 Creating the Mint

The mint chooses a prime, p, with $(p-1)/2$ also prime, a generator, g, s.t.

$$g^2 \neq 1 \pmod{p} \tag{1}$$

and

$$g^{(p-1)/2} = 1 \pmod{p} \tag{2}$$

(see 7.1) and a random number, k,

$$k \in [0, (p-1)/2) \tag{3}$$

Let G be the group generated by g.
The mint publishes

$$(g, p, g^k \pmod{p}) \tag{4}$$

2.2 Withdrawing a Coin

To withdraw a coin Alice picks a random x, the coin ID, from a sufficiently large set that two equal values are unlikely to ever be generated[2], and calculates,

$$y = \text{oneway}(x) \tag{5}$$

(see 7.2). y should be in G; check that

$$y^{(p-1)/2} = 1 \pmod{p} \tag{6}$$

however, note that the current form of the oneway function ensures that y is in G, so this check is redundant.

[1]At least, that's what people think. Take legal advice before using this stuff!

[2]Remember that if the size of the set of all possible coins is C, the probability of two being the same is .5 after around $\sqrt{C}$ coins have been generated.

Alice chooses a random blinding factor $b \in [0, (p-1)/2)$ and sends yg^b (the coin request) to the mint[3]. The mint debits Alice's account and returns the blinded signature,

$$m = (yg^b)^k \pmod{p} \tag{7}$$

Alice unblinds m, calculating the signature,

$$z = m(g^k)^{-b} = (yg^b)^k g^{-kb} = y^k g^{bk} g^{-kb} = y^k \pmod{p} \tag{8}$$

The coin is then

$$c = (x, z) \tag{9}$$

2.3 Spending a Coin

To spend a coin, Alice simply gives the coin, c, to Bob. Bob then sends it to the mint to be checked. The mint first ensures that x has not already been spent, and that $\text{oneway}(x)$ is in G, then checks that z is a signature for x (i.e. $z = \text{oneway}(x)^k \pmod{p}$). The mint then records x as spent and credits Bob's account.

3 Attack

Unfortunately an attack on the anonymity of this protocol is possible. The mint can mark a coin in a way that only it can detect, by signing it with k' instead of k. Then the unblinded "signature" is

$$z = (yg^b)^{k'} g^{-bk} = y^{k'} g^{b(k'-k)} \pmod{p} \tag{10}$$

When Bob submits c to the mint, then the mint calculates

$$y(zy^{-k'})^{1/(k'-k)} = y(g^{b(k'-k)})^{1/(k'-k)} = yg^b \pmod{p} \tag{11}$$

The mint can then simply look up who sent yg^b to it and thus learn Alice's identity.

4 Type I Defence

One defence against this attack is to make the mint prove that it has signed with k and not some other number. Since the mint must not reveal k, this proof must be a zero-knowledge proof. Two possible zero-knowledge proofs are known to me.

[3]Paulo Barreto points out that the efficiency of the scheme can be improved by calculating $g^{b.\text{preoneway}(x)} = yg^b$, thus saving an exponentiation.

4.1 Variation 1

This variation was suggested by Ian Goldberg.

Given a coin request, yg^b, the mint chooses a random number r s.t.

$$r \in [\log_g(p) + 1, (p-1)/2 - \log_g(p) - 1] \tag{12}$$

and calculates

$$t = k/r \pmod{(p-1)/2} \tag{13}$$

($(p-1)/2$ rather than p because we are working in G, which has order $(p-1)/2$). The mint then sends Alice

$$Q = (yg^b)^r \pmod{p} \tag{14}$$

and

$$A = g^r \pmod{p} \tag{15}$$

Alice then randomly demands one of r or t.

If Alice chose r, she verifies that

$$Q = (yg^b)^r \pmod{p} \tag{16}$$

and

$$A = g^r \pmod{p} \tag{17}$$

If Alice chose t, she verifies that

$$A^t = g^{rt} = g^k \pmod{p} \tag{18}$$

and

$$Q^t = (yg^b)^{rt} = (yg^b)^k = z \pmod{p} \tag{19}$$

Note that a mint that wants to cheat has a .5 chance of getting away with it each time (by guessing whether the challenger will choose r or t and lying about Q and A appropriately). Naturally, it is increasingly unlikely to get away with this with each repetition. A suspicious challenger could always repeat the protocol until the probability of cheating is low enough to make them happy.

4.2 Variation 2

This variation is due to Chaum and Pedersen (Crypto '92) (I'm told).

The mint chooses a random value r and sends Alice

$$u = g^r \pmod{p} \tag{20}$$

and

$$v = (yg^b)^r \pmod{p} \tag{21}$$

Alice responds with a challenge d. The mint answers with

$$w = dk + r \pmod{(p-1)/2} \tag{22}$$

Alice verifies that

$$g^w = g^{dk+r} = (g^k)^d u \pmod{p} \tag{23}$$

and

$$(yg^b)^w = (yg^b)^{dk+r} = ((yg^b)^k)^d v = (yg^b)^d v \pmod{p} \tag{24}$$

4.3 Non-interactive variant

It is suggested that choosing

$$d = hash(u, v) \tag{25}$$

would allow the second variation to be used non-interactively. The mint sends (d, w) along with the coin, Alice calculates

$$g^w (g^k)^{-d} = u \pmod{p} \tag{26}$$

and

$$(yg^b)^w S^{-d} = v \pmod{p} \tag{27}$$

and verifies that $d = hash(u, v)$.

I'm not entirely convinced that it isn't possible to search for (or even calculate) a set of values that makes this appear to work whilst still signing with k'.

5 Type II Defence

Another defence is to combine two blinding methods, using two indepenent random blinding factors. With this method, the coin-withdrawal protocol changes as follows.

To withdraw a coin Alice picks a random x, the coin ID, from a sufficiently large set that two equal values are unlikely to ever be generated, and calculates,

$$y = \text{oneway}(x) \tag{28}$$

(see 7.2). y should be in G; check that

$$y^{(p-1)/2} = 1 \pmod{p} \tag{29}$$

Alice chooses random blinding factors $b_y, b_g \in [0, (p-1)/2)$, ensuring that b_y is invertible $mod(p-1)/2$ and sends $y^{b_y} g^{b_g}$ (the coin request) to the mint. The mint debits Alice's account and returns the blinded signature,

$$m = (y^{b_y} g^{b_g})^k \pmod{p} \tag{30}$$

Alice unblinds m, calculating the signature,

$$\begin{aligned} z &= (m.(g^k)^{-b_g})^{1/b_y} && (31)\\ &= ((y^{b_y} g^{b_g})^k g^{-kb_g})^{1/b_y} && (32)\\ &= (y^{kb_y} g^{kb_g} g^{-kb_g})^{1/b_y} && (33)\\ &= (y^{kb_y})^{1/b_y} && (34)\\ &= y^k \pmod{p} && (35) \end{aligned}$$

Now z is in the same form as in the original scheme and we can proceed as normal.

5.1 Failed Attack

If the mint attempts to mark the coin, as before, then let's see what happens. The blinded signature is

$$m = (y^{b_y} g^{b_g})^{k'} \pmod{p} \tag{36}$$

unblinding, Alice gets

$$\begin{aligned} z &= (m.(g^k)^{-b_g})^{1/b_y} && (37)\\ &= ((y^{b_y} g^{b_g})^{k'} g^{-kb_g})^{1/b_y} && (38)\\ &= (y^{k'b_y} g^{k'b_g} g^{-kb_g})^{1/b_y} && (39)\\ &= (y^{k'b_y} g^{(k'-k)b_g})^{1/b_y} && (40)\\ &= y^{k'} g^{(k'-k)b_g/b_y} \pmod{p} && (41) \end{aligned}$$

Because this result entangles both the unknown (to the mint) value y and the, also unknown, value g^{b_g/b_y}, the mint cannot even verify that this is a correct signature, let alone figure out who gave it the blinded coin in the first place.

6 Cost and Value

Although there are those that hold that a coin should have a value similar to its cost of production, this is clearly insane, at least when the coin is to be used as money[4].

In general, the cost of production should be considerably less than the value of the coin. So, it is worth calculating the cost of producing Lucre coins.

Assuming that the coins are relatively low value, then a 512 bit signing key should be sufficient. The cost of producing a coin is really the cost of signing it twice (once blinded when withdrawn, and once ublinded when deposited). Implemented in Java on a 300 MHz Pentium[5] we can achieve 25 signs per second. A server in the Bunker (http://www.thebunker.net/) costs £250 per month.

That's £8 per day. 30p per hour, .5p per minute, .001p per second, .0004p per sign.

So, values of .01p per coin are easily achievable.

Incidentally, signing with a 1024-bit key takes around 6 times as long, so values of .1p with 1024-bit security are also achievable.

7 Theory

7.1 Subgroup Order

(2) ensures that the order of the subgroup generated by g is $(p-1)/2$.

7.1.1 Leakage

This avoids leakage of information about k which can occur if g generates the whole of Z_p^*, because

$$(g^k)^{(p-1)/2} \begin{cases} = 1 & \text{if } k \text{ is even} \\ \neq 1 & \text{if } k \text{ is odd} \end{cases} \tag{42}$$

Proof

If k is even, then there exists an n s.t. $k = 2n$.

$$(g^{2n})^{(p-1)/2} = (g^n)^{p-1} \tag{43}$$

[4]A clear example where it is not insane is Adam Back's hashcash used as an anti-spam measure - in that case, the whole point is that the coin is expensive to produce.

[5]Surely nothing can be slower that this?

Since

$$gcd(g^n, p) = 1 \tag{44}$$

then, by Euler's theorem,

$$(g^n)^{p-1} = 1 \pmod{p} \tag{45}$$

If k is odd, then there exists an n s.t. $k = 2n + 1$.

$$(g^{2n+1})^{(p-1)/2} = (g^n)^{p-1} g^{(p-1)/2} \tag{46}$$

$$(g^n)^{p-1} = 1 \pmod{p} \tag{47}$$

(see (45)) and

$$g^{(p-1)/2} \neq 1 \pmod{p} \tag{48}$$

because the order of g is $p - 1$, so no $y < p - 1$ can give $g^y = 1 \pmod{p}$. So

$$(g^n)^{p-1} g^{(p-1)/2} = 1 \cdot x \pmod{p}, x \neq 1 \tag{49}$$

7.1.2 Invertability

The ZK proofs require exponents to be invertible, and in any case this may be a useful property. This would not be possible in an exponent group of order $p - 1$ because $x^{-1} \pmod{p-1}$ does not exist if $gcd(x, p - 1) \neq 1$, which would be the case for all even x.

7.1.3 Subgroup Order Revisited

It has been pointed out that using a g that generates the whole group Z_p^* and choosing k odd also fixes both the above problems, and makes some parts of the protocol cheaper (because you can avoid the exponentiation in the one-way function). This seems to me to be somehow less satisfying, but I can't see anything actively wrong with it.

7.2 One-way Coin Function

The purpose of the one way function is to prevent Alice from cheating the mint by producing variants on a signed coin by simpy reblinding the coin and the signature - the fact that the coin has a special structure prevents this from working.

The one-way coin function can, in principle, be any one way function, but the one chosen for Lucre is defined as follows: Let the random seed for the coin be in $[0, 2^n)$ where

$$n = m + ((\log_2(p) - m) \bmod 160) \tag{50}$$

m is the minimim number of bits in x, chosen to be large enough to avoid collisions (128 in Lucre's case). Then define

$$h_0(x) = x, h_k(x) = h_{k-1}(x)|SHA1(h_{k-1}(x)) \tag{51}$$

where | denotes concatenation. Then

$$\text{preoneway}(x) = h_{(n-m)/160}(x) \tag{52}$$

In case it isn't obvious, this ensures that

$$\log_2(\text{preoneway}(x)) \approx \log_2(p) \tag{53}$$

We then ensure that oneway(x) is in G

$$\text{oneway}(x) = g^{\text{preoneway}(x)} \pmod p \tag{54}$$