Lucre: Anonymous Electronic Tokens

Ben Laurie
ben@algroup.co.uk

November 30, 1999

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 and Anonymous.

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 5.1) and a random number, k,

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

The ends are chopped to off to avoid an attack to find the discrete log by calculating the standard (i.e. non-discrete) log.

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, and calculates,

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

(see 5.2). y should be in G; check that

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

Alice chooses a random blinding factor b and sends yg^b (the coin request) to the mint. The mint debits Alice's account and returns the blinded signature,

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

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

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 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 Defence

The 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.

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

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

.

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 c. The mint answers with

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

Alice verifies that

$$g^w = g^{ck+r} = (g^k)^c u \,(\mathrm{mod}\, p) \tag{23}$$

and

$$(yg^b)^w = (yg^b)^{ck+r} = ((yg^b)^k)^c v = (yg^b)^c v \,(\mathrm{mod}\, p) \tag{24}$$

4.3 Non-interactive variant

It is suggested that choosing

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

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

$$g^w (g^k)^{-c} = u \,(\mathrm{mod}\, p) \tag{26}$$

and

$$(yg^b)^w S^{-c} = v \,(\mathrm{mod}\, p) \tag{27}$$

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

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 Theory

5.1 Subgroup Order

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

5.1.1 Leakage

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

$$(g^k)^{(p-1)/2} \begin{cases} = 1 & \text{if } k \text{ is even} \\ \neq 1 & \text{if } k \text{ is odd} \end{cases} \tag{28}$$

Proof

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

$$(g^{2n})^{(p-1)/2} = (g^n)^{p-1} \tag{29}$$

Since

$$gcd(g^n, p) = 1 \tag{30}$$

then, by Euler's theorem,

$$(g^n)^{p-1} = 1 \pmod p \tag{31}$$

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

$$(g^{2n+1})^{(p-1)/2} = (g^n)^{p-1} g^{(p-1)/2} \tag{32}$$

$$(g^n)^{p-1} = 1 \pmod p \tag{33}$$

(see (31)) and

$$g^{(p-1)/2} \neq 1 \pmod p \tag{34}$$

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

$$(g^n)^{p-1} g^{(p-1)/2} = 1 \cdot x \pmod p, x \neq 1 \tag{35}$$

5.1.2 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.

5.2 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

$$n = m + ((\log_g(p) - m) \bmod 160) \tag{36}$$

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

$$h_0(x) = x, h_k(x) = h_{k-1}(x) | SHA1(h_{k-1}(x)) \tag{37}$$

where | denotes concatenation. Then

$$\text{oneway}(x) = h_{(n-m)/160}(x) \tag{38}$$

In case it isn't obvious, this ensures that

$$\log_g(\text{oneway}(x)) \approx \log_g(p) \tag{39}$$