Bit gold

6-7 minutes

A long time ago I hit upon the idea of bit gold. The problem, in a nutshell, is that our money currently depends on trust in a third party for its value. As many inflationary and hyperinflationary episodes during the 20th century demonstrated, this is not an ideal state of affairs. Similarly, private bank note issue, while it had various advantages as well as disadvantages, similarly depended on a trusted third party.

Precious metals and collectibles have an unforgeable scarcity due to the costliness of their creation. This once provided money the value of which was largely independent of any trusted third party. Precious metals have problems, however. It's too costly to assay metals repeatedly for common transactions. Thus a trusted third party (usually associated with a tax collector who accepted the coins as payment) was invoked to stamp a standard amount of the metal into a coin. Transporting large values of metal can be a rather insecure affair, as the British found when transporting gold across a U-boat infested Atlantic to Canada during World War I to support their gold standard. What's worse, you can't pay online with metal.

Thus, it would be very nice if there were a protocol whereby unforgeably costly bits could be created online with minimal dependence on trusted third parties, and then securely stored, transferred, and assayed with similar minimal trust. Bit gold.

My proposal for bit gold is based on computing a string of bits from a string of challenge bits, using functions called variously "client puzzle function," "proof of work function," or "secure benchmark function.". The resulting string of bits is the proof of work. Where a one-way function is prohibitively difficult to compute backwards, a secure benchmark function ideally comes with a specific cost, measured in compute cycles, to compute backwards.

Here are the main steps of the bit gold system that I envision:

(1) A public string of bits, the "challenge string," is created (see step 5).

(2) Alice on her computer generates the proof of work string from the challenge bits using a benchmark function.

(3) The proof of work is securely timestamped. This should work in a distributed fashion, with several different timestamp services so that no particular timestamp service need be substantially relied on.

(4) Alice adds the challenge string and the timestamped proof of work string to a distributed property title registryfor bit gold. Here, too, no single server is substantially relied on to properly operate the registry.

(5) The last-created string of bit gold provides the challenge bits for the next-created string.

(6) To verify that Alice is the owner of a particular string of bit gold, Bob checks the unforgeable chain of title in the bit gold title registry.

(7) To assay the value of a string of bit gold, Bob checks and verifies the challenge bits, the proof of work string, and the timestamp.

Note that Alice's control over her bit gold does not depend on her sole possession of the bits, but rather on her lead position in the unforgeable chain of title (chain of digital signatures) in the title registry.

All of this can be automated by software. The main limits to the security of the scheme are how well trust can be distributed in steps (3) and (4), and the problem of machine architecture which will be discussed below.

Hal Finney has implemented a variant of bit gold called RPOW (Reusable Proofs of Work). This relies on publishing the computer code for the "mint," which runs on a remote tamper-evident computer. The purchaser of of bit gold can then use remote attestation, which Finney calls the transparent server technique, to verify that a particular number of cycles were actually performed.

The main problem with all these schemes is that proof of work schemes depend on computer architecture, not just an abstract mathematics based on an abstract "compute cycle." (I wrote about this obscurely several years ago.) Thus, it might be possible to be a very low cost producer (by several orders of magnitude) and swamp the market with bit gold.

However, since bit gold is timestamped, the time created as well as the mathematical difficulty of the work can be automatically proven. From this, it can usually be inferred what the cost of producing during that time period was.

Unlike fungible atoms of gold, but as with collector's items, a large supply during a given time period will drive down the value of those particular items. In this respect "bit gold" acts more like collector's items than like gold. However, the match between this ex post market and the auction determining the initial value might create a very substantial profit for the "bit gold miner" who invents and deploys an optimized computer architecture.

Thus, bit gold will not be fungible based on a simple function of, for example, the length of the string. Instead, to create fungible units dealers will have to combine different-valued pieces of bit gold into larger units of approximately equal value. This is analogous to what many commodity dealers do today to make commodity markets possible. Trust is still distributed because the estimated values of such bundles can be independently verified by many other parties in a largely or entirely automated fashion.

In summary, all money mankind has ever used has been insecure in one way or another. This insecurity has been manifested in a wide variety of ways, from counterfeiting to theft, but the most pernicious of which has probably been inflation. Bit gold may provide us with a money of unprecedented security from these dangers. The potential for initially hidden supply gluts due to hidden innovations in machine architecture is a potential flaw in bit gold, or at least an imperfection which the initial auctions and ex post exchanges of bit gold will have to address.