Class 6: Proofs of Work

Schedule

- If you didn't get full credit for Project 1 because of failure to post something interesting, you can (and should!) redeem yourself and earn full credit by **posting an interesting comment by Thursday**. It can be on (1) Discussion questions from Project 1 (2) notes from classes, or (3) general forum.
- **Read:** *Chapter 6: The Bitcoin Network*, *Chapter 7: The Blockchain* from Andreas Antonopoulos' book. (Ideally, you should finish these before Wednesday's class, but at the latest by Monday, 9 Feb.)

Trust

What are valid sources of *trust*?

What are invalid sources of *trust*?

What mechanisms have humans evolved or constructed to enhance trust among strangers?

Distributed Consensus

How well does the 2-out-of-3 network consensus public ledger protocol work?

Proof-of-Work

Cynthia Dwork and Moni Naor. *Pricing via Processing or Combatting Junk Mail*, CRYPTO 1992.

Pricing Function: (f) - moderately easy to compute - cannot be amortized - computing $f(m_1), ..., f(m_l)$ costs l times as much as computing $f(m_i)$. - easily verified: given x, y easy to check $y = f(x)$.

Adam Back. *Hash Cash Postage Implementation*

Interactive Hashcash:
1. Sender to Receiver: `Hello`
2. Receiver to Sender: r (random nonce)
3. Sender to Receiver: x, `Mail` where $x = f(r)$
4. Receiver verifies $x = f(r)$.

How well does this protocol work for sending mail?

Suppose we use SHA-256 for f?

How can we make this protocol non-interactive?

Bitcoin Mining

Proof-of-work: Find an x such that: SHA-256(SHA-256($r + x$)) $< T/d$.

d is the "difficulty" (varies).
T is a fixed target (256-bit number).
r depends on hash of previous block, transactions, and other information.

What does it mean for the bitcoin difficulty to go down?

BitcoinMiner (code from core Bitcoin implementation)

```
//
// ScanHash scans nonces looking for a hash with at least some zero bits.
// The nonce is usually preserved between calls, but periodically or if the
// nonce is 0xffff0000 or above, the block is rebuilt and nNonce starts over at
// zero.
//
bool static ScanHash(const CBlockHeader *pblock, uint32_t& nNonce, uint256 *phash)
{
    // Write the first 76 bytes of the block header to a double-SHA256 state.
    CHash256 hasher;
    CDataStream ss(SER_NETWORK, PROTOCOL_VERSION);
    ss << *pblock;
    assert(ss.size() == 80);
    hasher.Write((unsigned char*)&ss[0], 76);

    while (true) {
        nNonce++;

        // Write the last 4 bytes of the block header (the nonce) to a copy of
        // the double-SHA256 state, and compute the result.
        CHash256(hasher).Write((unsigned char*)&nNonce, 4).Finalize((unsigned char*)phash);

        // Return the nonce if the hash has at least some zero bits,
        // caller will check if it has enough to reach the target
        if (((uint16_t*)phash)[15] == 0)
            return true;

        // If nothing found after trying for a while, return -1
        if ((nNonce & 0xfff) == 0)
            return false;
    }
}
```

BitcoinMiner: (excerpted, most error checking code removed)

```
void static BitcoinMiner(CWallet *pwallet)
{
    SetThreadPriority(THREAD_PRIORITY_LOWEST);
    CReserveKey reservekey(pwallet);
    unsigned int nExtraNonce = 0;

    try {
        while (true) {
            // Create new block
            unsigned int nTransactionsUpdatedLast = mempool.GetTransactionsUpdated();
```

```
            CBlockIndex* pindexPrev = chainActive.Tip();

            auto_ptr<CBlockTemplate> pblocktemplate(CreateNewBlockWithKey(reservekey));
            CBlock *pblock = &pblocktemplate->block;
            IncrementExtraNonce(pblock, pindexPrev, nExtraNonce);

            int64_t nStart = GetTime();
            arith_uint256 hashTarget = arith_uint256().SetCompact(pblock->nBits);
            uint256 hash;
            uint32_t nNonce = 0;
            while (true) {
                // Check if something found
                if (ScanHash(pblock, nNonce, &hash))
                {
                    if (UintToArith256(hash) <= hashTarget)
                    {
                        // Found a solution
                        pblock->nNonce = nNonce;
                        assert(hash == pblock->GetHash());

                        SetThreadPriority(THREAD_PRIORITY_NORMAL);
                        LogPrintf("proof-of-work found  \n  hash: %s  \ntarget: %s\n",
                              hash.GetHex(), hashTarget.GetHex());
                        ProcessBlockFound(pblock, *pwallet, reservekey);
                        SetThreadPriority(THREAD_PRIORITY_LOWEST);
                        break;
                    }
                }

                if (nNonce >= 0xffff0000) break;
                // ... other breaking conditions elided
                // Update nTime every few seconds
                UpdateTime(pblock, pindexPrev);
            }
        }
    }
    catch (const boost::thread_interrupted&)
    {
        LogPrintf("BitcoinMiner terminated\n");
        throw;
    }
}
```