Crypto 6: Passwords & Blockchains

YOU DON'T GET TO CHOSE YOUR THREAT MODEL
YOU YOUR THREATS GET TO CHOSE YOU

- Lea Kissner

THE ONLY GOOD BLOCKCHAIN
IS HOMER SIMPSON'S BLOCKCHAIN
Reminder:
Cryptographic Hashes...

- We love ourselves some cryptographic hashes
  - SHA_256, SHA_384, SHA3_256, SHA3_384

- Reminder on the properties:
  - Irreversible:
    Given $H(X)$, it is infeasible to find $X$ short of simply trying all possibilities
  - First preimage resistant:
    Given $H(X)$, it is infeasible to find any $X'$ such that $H(X) = H(X')$
  - Second preimage resistant:
    It is infeasible to find $X$ and $Y$ such that $X \neq Y$ and $H(X) = H(Y)$
A Couple Other Hash Properties...

- They accept arbitrarily large inputs
- They "look" random
  - Change a single bit on the input and each output bit has a 50% chance of flipping
  - And until you change the input, you can't predict which output bits are going to change
- The ones we talked about are **fast**
  - Can operate at many many MB/s:
    Faster at processing data than block ciphers
A Hash Problem: Proof of work...

- Alice wants Bob to waste a bunch of CPU resources
  - But wants to quickly **check** that Bob wasted that much CPU
- Alice -> Bob: "Here is a message $M$ and a factor $x$"
  - Make sure $M$ has a nonce in it
- Now Bob needs to provide $M'$ such that it starts with $M$ and $H(M')$ starts with $x$ zero bits
- Alice computes $H(M')$ and verifies that it starts with $x$ zero bits
  - Alice now knows that Bob is expected to have had to create $2^x$ separate $M$'s and hash them until he found one that matched
What this provides

- You can use it in a protocol where the user has to waste something...
  - EG, proposals for sending mail as a way of reducing spam
    - It wouldn't: Bad guys can get lots of CPU resources
- Have other options too
- CAPTCHAs:
  - Those "prove your human" web puzzles:
    - It is a proof you wasted a few seconds of a human's time
    - (Or that you paid $.01 to waste a few seconds of a human's time)
- Proof of \textit{wait}
  - Alice has a secret key $k$
  - Alice to Bob sends "Don't contact me until time $T$, here is HMAC($k,T$)"
  - When Bob gets back, he says "$T$, HMAC($k,T$)"
  - Alice then verifies $T$ is in the past and HMAC($k,T$)
Passwords

- The password problem:
  - User Alice authenticates herself with a password $P$
- How does the site verify later that Alice knows $P$?
- Classic:
  - Just store \{Alice, $P$\} in a file...
- But what happens when the site is hacked?
  - The attacker now knows Alice's password!
- Enter "Password Hashing"
Password Hashing

• Instead of storing \{Alice, P\}...
  • Store \{Alice, H(P)\}

• To verify Alice, when she presents P
  • Compute $H(P)$ and compare it with the stored value

• Problem: Brute Force tables...
  • Most people chose bad passwords...
    And these passwords are known
  • Bad guy has a huge file...
    • $H(P1), P1$
    • $H(P2), P2$
    • $H(P3), P3$...
  • Ways to make this more efficient ("Rainbow Tables")
A Sprinkle of Salt...

• Instead of storing \{Alice, H(P)\}, also have a user-specific string, the "Salt"
  • Now store \{Alice, Salt, H(P||Salt)\}
  • The salt ideally should be both long and random, but it isn't considered "secret": rather it is a nonce

• As long as the salt is unique...
  • An attacker who captures the password file has to brute force Alice's password on its own
  • Its still an "off-line attack" (Attacker can do all the computation he wants) but...
    • At least the attacker can't precompute possible solutions
Slower Hashes...

- Most cryptographic hashes are designed to be *fast*
  - After all, that is the point: they should not only turn $H(🐮)$ to hamburger... they need to do it quickly

- But for password hashes, we *want* it to be slow!
  - Its OK if it takes a good fraction of a second to check a password
    - Since you only need to do it once for each legitimate usage of that password
    - But the attacker needs to do it for each password he wants to try

- Slower hashes don't change the *asymptotic difficulty* of password cracking but can have huge practical impact
  - Slow rate by a factor of 10,000 or more!


**PBKDF2**

- "Password Based Key Derivation Function 2"
  - Designed to produce a long "random" bitstream derived from the password

- Used for both a password hash and to generate keys derived from a user's password

- PBKDF(PRF, P, S, c, len):
  - **PRF** == Pseudo Random Function (e.g. HMAC-SHA256)
  - **P** == Password
  - **S** == Salt
  - **c** == Iteration count
  - **len** == Number of bits/bytes requested
  - **DK** == Derived Key

```python
PBKDF(PRF, P, S, c, len):  
DK = ""  
for i = 1, range(len/blocksize)+1) {  
    DK = DK || F(PRF, P, S, c, i)  
}  
return DK[0:len]
```

```python
F(PRF, P, S, c, i) {  
    UR = U = PRF(P, S||INT_32(i))  
    for j = 2; j <= c; ++j {  
        U = PRF(P, U)  
        UR = UR ^ U  
    }  
    return UR
}
```
Comments on PBKDF2

• Allows you to get effectively an arbitrary long string from a password
  • Assuming the user's password is strong/high entropy

• Very good for getting a bunch of symmetric keys from a single password
  • You can also use this to seed a pRNG for generating a "random" public/private key pair

• Designed to be slow in computation...
  • But it does not require a lot of memory:
    Other functions are also expensive in memory as well, e.g. scrypt & argon2
Passwords...

- If an attacker can do an **offline** attack, your password must be **really good**
  - Attacker simply tries a huge number of passwords in parallel using a GPU-based computer: buy a bunch of used Nvidia 2080 supers from all those upgrading to 3080s
  - So you need a **high entropy** password:
    - Even xkcd-style is only 10b/word with a 1000 word dictionary, so need a 7 or more **random word** passphrase to resist a determined attacker
- Life is far better is if the attacker can only do **online** attacks:
  - Query the device and see if it works
  - Now limited to a few tries per second and **no parallelism**!
... and iPhones

- Apple's security philosophy:
  - In your hands, the phone should be everything
  - In anybody else's, it should (ideally) be an inert "brick"

- Apple uses a small co-processor in the phone to handle the cryptography
  - The "Secure Enclave"

- The rest of the phone is untrusted
  - Notably the memory: *All* data must be encrypted:
    The CPU requests that the Secure Enclave unencrypt data and some data (e.g., your credit card for ApplePay) is only readable by the Secure Enclave

- They also have an ability to effectively erase a small piece of memory
  - "Effaceable Storage": this takes a good amount of EE trickery
Crypto and the iPhone Filesystem

- A lot of keys encrypted by keys...
  - But there is a random master key, $k_{\text{phone}}$, that is the root of all the other keys
- Need to store $k_{\text{phone}}$ encrypted by the user's password in the flash memory
  - $\text{PBKDF2}(P,...) = k_{\text{user}}$
- But how to prevent an off-line brute-force attack?
  - Also have a 256b random secret burned into the Secure Enclave that you can use for encryption
    - Need to take apart the chip to get this!
  - Even the secure enclave can't read this secret, only use this secret as a key for hardware cryptographic engines
- Now the user key is not just a function of $P$, but $E(K_{\text{secret}}, P)$
  - Without the secret, can not do an offline attack
- All online attacks have to go through the secure enclave
  - After 5 tries, starts to slow down
  - After 10 tries, can (optionally) nuke $k_{\text{phone}}$!
    - Erase just that part of memory -> effectively erases the entire phone!
    - Even compromising the secure enclave limits guessing to 10 per second!
Backups...

• Of course there is a **necessary** weakness:
  • Backing up the phone copies all the data off in a form not encrypted using the in-chip secret
    • After all, you need to be able to recover it onto a new phone!

• So someone who can get your phone...
  And can somehow managed to have it unlocked
  • Thief, abusive boyfriend, cop...
    • Hold it up to your face (iPhone X) or Fingerprint (5s or beyond)
    • And then sync it with a new computer

• Change of policy for iOS-11:
  • Now you also need to put in the passcode to trust a new computer:
    Can't create a backup without knowing the passcode
Why Talk About Cryptocurrencies?!?

• I am an actual **expert** in this area
  • It has been one of my research focuses for the past 8+ years!

• But I want it to die in a fire!
  • There is effectively no value:
    • Private Blockchains are 20+ year old ideas
    • Public Blockchains are grossly inefficient in the name of "decentralization" without actually being decentralized!
      • And don't actually solve any problems other than those required to implement cryptocurrencies!
    • Cryptocurrencies don't work as currency unless you are a criminal!

• Yet it has refused to just go away
  • And it touches on a lot of real world "security" issues that often have nothing to do with actual security!
Linked Lists Blockchains And CryptoCurrencies

• “Blockchain Technology”
  • A fancy word for “Append-Only Data Structure”
    • That causes people’s eyes to glaze over and them to throw money at people
  • “Private/Permissioned Blockchain”:
    • A setup where only one or a limited number of systems are authorized to append to the log
    • AKA 20 year old, well known techniques
  • “Public/Permissionless Blockchain”:
    • Anybody can participate as appenders so there is supposedly no central authority:
      Difficulty comes in removing “sibyls”

• Cryptocurrencies
  • Things that don’t actually work as currencies…
Hash Chains

- If a data structure includes a hash of the previous block of data: This forms a “hash chain”
- So if you have a way of validating the ending block: The inclusion of the previous block’s hash validates all the previous blocks
- This also makes it easy to add blocks to data structures
  - Only need to hash block + hash of previous block, rather than rehash everything:
    How you can efficiently hash an "append only" datastructure
- Now just validate the head (e.g. with signatures) and voila!
  - All a “blockchain” is is a renamed hashchain!
    Linked timestamping services used this structure and were proposed back in 1990!
  - Certificate Revocation Lists are signed hash-chains
Merkle Trees

- Lets say you have a lot of elements
  - And you want to add or modify elements
- And you want to make the hash of the set easy to update
- Enter hash trees/merkle trees
  - Elements 0, 1, 2, 3, 4, 5...
  - $H(0)$, $H(1)$, $H(2)$...
  - $H(H(0) \oplus H(1))$, $H(H(2)\oplus H(3))$...
  - The final hash is the root of the top of the tree.
- And so on until you get to the root
  - Allows you to add an element and update $\log(n)$ hashes
    Rather than having to rehash all the data
- Patented in 1979!!
A Trivial Private Blockchain…

- We have a single server $s$, with keys $K_{pub}$ and $K_{priv}$…
  - And a git archive $g$… (in which we fix git to use SHA-256)

- Whenever we issue a pull request…
  - The server validates that the pull request meets the allowed criteria
  - Accepts the pull request
  - Signs the hash of the head…

- And that is it!
  - Git is an append only data structure, and by signing the new head, we have the server authenticating the *entire archive*!

- This is why “private” blockchain is *not* a revolution!!!
  - Anything that would benefit from an append-only, limited writer database already has one!
What Is A "Cryptocurrency"?

- A cryptocurrency is a tradable cryptographic token
  - The goal is to create irreversible electronic cash with no centralized trust: If Alice wants to pay Bob 200 Quatloos to pay off her losing bet on the Green thrall, there should be nobody else who can block or reverse this transfer

- Based on the notion of a public ledger (the "Blockchain")
  - A public shared document that says "Alice has 3021.1141 Quatloos, Bob has 21.13710 Quatloos, Carol has 1028.8120 Quatloos..."
  - People can only add items to the ledger ("append-only"), never remove items

- Big Idea: Alice writes and signs a check to Bob saying "I, Alice, Pay Bob 200 Quatloos"
  - This check then gets added to the public ledger so now everyone knows Alice now has 2821.1141 Quatloos and Bob has 221.13710 Quatloos
What Is A "Cryptocurrency"?
What Is A "Blockchain"
(well, "Public" or "Permissionless" Blockchains)

- Everyone involved gathers up copies of the loose checks
  - For each check, validate that there are sufficient funds
  - Bundle all the checks up into a "block" and staple them together, with a pointer to the previous pile
- Everybody now does a lot of useless "work" that may eventually get lucky
  - The one that gets lucky staples this (which is in the form of a check saying "The system pays to ME the reward for success" and the staple that binds everything together) to the block as well, publishes this, and gets the reward
- Now everybody else knows this stapled pile of checks is now verified
  - So everybody starts on a new block, pointing to the previous block and gathers up the new checks that haven't yet been processed
- Result is an **append only** data structure
What Is A "Blockchain" (well, "Public" or "Permissionless" Blockchains)
What Is Bitcoin?

• Simply the first widespread development of this idea
  • A "Bitcoin wallet" is simply a collection of cryptographic keys
    • Private key $K_{\text{priv}}$: A secret value stored in the wallet
    • Public key $K_{\text{pub}}$: A public value that anybody is allowed to see, derived from the private key
  • The "Bitcoin Blockchain" is Bitcoin's particular implementation of the shared ledger

• Spending Bitcoin is simply writing a check and broadcasting it:
  • "Pay $K_{\text{pub}}=1\text{Ross5Np5doy4ajF9iGXzgKaC2Q3Pwwxv}$ the value 0.05212115 Bitcoin...
    And whoever validates this transaction gets 0.0005 Bitcoin"
  - Signed 1FuckBTCqwBQexxs9jiuWTiZeoKfSo9Vyi:
  • This is Bitcoin transaction
d6b24ab29fa8e8f2c43bb07a3437538507776a671d9301368b1a7a32107b7139
What Is Bitcoin?

- Simply the first widespread development of this idea
- A "Bitcoin wallet" is simply a collection of cryptographic keys
  - Private key $K_{priv}$: A secret value stored in the wallet
  - Public key $K_{pub}$: A public value that anybody is allowed to see, derived from the private key
    - Public key signature: A function pair:
      - $\text{Sign}(X,K_{priv}) \rightarrow Y$ (can only be done if you know $K_{priv}$)
      - $\text{Verify}(X,Y,K_{pub}) \rightarrow \text{True/False}$ (can be done by anybody who knows $K_{pub}$)
- The "Bitcoin Blockchain" is Bitcoin's particular implementation of the shared ledger
- Spending Bitcoin is simply writing a check and broadcasting it:
  - "Pay $K_{pub}$=1Ross5Np5doy4ajF9iGXzgKaC2Q3Pwwxv the value 0.05212115 Bitcoin...
  - And whoever validates this transaction gets 0.0005 Bitcoin"
- Signed 1FuckBTCqwBQexxs9jiuWTIZecKfSo9Vyi:
  - This is Bitcoin transaction d6b24ab29fa9f2c43bb07a3437538507776a671d9301368b1a7a32107b7139
  - 1Ross5Np5doy4… (Free Ross Ulbricht !?) - (Spent) 0.05212115 BTC
  - 0.05212115 BTC
What Is Bitcoin Mining?

- It is the particular instance used to protect the transaction history for Bitcoin
  - Based on SHA-256
- Every miner takes all the unconfirmed transactions and puts them into a block
  - The block has fixed capacity (currently 1MB), limiting the global rate to ~3-7 transactions per second, and also includes a timestamp
  - Also attaches the "pay me the block reward and all fees" check to the front (the "coinbase")
  - Also attaches the hash of the previous block (including by reference everything in the past)
- Then performs the "Proof of work" calculation
  - Just hashes the block, changing it trivially until the hash starts with enough 0s.
    - This is the "difficulty factor", which automatically adjusts to ensure that, worldwide, a new block is discovered roughly every 10 minutes
- On success it broadcasts the new block
So Proof of Work...

- Remember, SHA256 looks random...
  - So just tweak one bit and the output looks totally different
- So if I present to you a string and the corresponding hash that starts with $n$ 0-bits...
  - I probably had to do $2^n$ hashes
- So you can trivially verify that I did a ton of useless work with just a single hash
The Blockchain Size Problem

- In order to verify that Alice has a balance...
  - You have to potentially check *every transaction* back to the beginning of the chain
- Results in amazingly inefficient storage
  - Every full Bitcoin node needs access to the *entire* transaction history
    - Because the entire history is needed to validate the transaction
  - A "lightweight" node still needs to keep the headers for all history
    - And still has to ask for suitable information to verify each transaction it needs to verify
- So if we have 10,000 nodes, this means 10,000 copies of the Bitcoin Blockchain!
Corollary: The Blockchain Capacity Problem...

- To limit the blockchain growth to "just" 1 MB a block...
  - An early defense against possible spam
  - The resulting design for Bitcoin can only process 3-7 transactions per second *worldwide*!
- Which means any "Bitcoin takes over money" requires trusted, centralized entities that maintain databases...
  - Oh, yeah, those are called banks! We have "electronic money" as a result, and have had it for decades!
- Also results in price shocks
  - When desired transactions < block capacity, transactions are cheap
  - When desired transactions > block capacity, prices spiral up because of an inelastic supply
  - Unknown attacks have cause transaction price shocks *for the lulz*!
The Blockchain Power Problem

- The Bitcoin system consumes roughly 8 GW of power right now (or basically Austria!)
- This is because Proof of Work creates a Red Queen's Race
  - As long as there is potential profit to be had you get an increase in capability
  - Efficiency gains get translated into more effort, not less power consumption: 10x the hashes doesn't mean 10x the bitcoin but just 10x the difficulty factor
- There is no way to reduce Bitcoin's power consumption without reducing Bitcoin's price or the block reward
  - It is this waste of energy that protects Bitcoin!
The Bitcoin Folks *lie* about the power consumption...

- Claim this rescues "stranded power"
  - But this is the point of a power *grid*: We ship electricity half-way across the country
    (Well, not to Texas because they have a separate grid so they can ignore federal regulations)
- Claim this incentivizes "green power"
  - But bitcoin mining wants 24/7/365 power ("base load")
    Base load power is only hydroelectric, fossil-fuel, or nuclear
  - And there really are no new spots for dams
- Oh, but other things burn power too...
  - Yeah, ALL data centers together is probably 2x-3x Bitcoin...
    But Bitcoin can only do 3-7 transactions per second on a WORLDWIDE BASIS!
  - And unlike Bitcoin, data centers try to reduce the power consumption
- Tesla's $1.5B is really a $1.5B in "Destroy the Planet Inc"
  Annual Bitcoin CO₂ emission of ~35 Mt of CO₂ is equivalent to driving an F150 Raptor for >60 billion miles!
The Sybil Problem...

- There is a lot of talk about "consensus" algorithms in cryptocurrencies
  - How the system agrees on a common view of history
  - Bitcoin's is simple: "Longest Chain Wins"

- But Proof of Work is **not** about consensus:
  - It is about solving the sybil (fake node) problem...
    - How do you prevent someone from just spinning up a gazillion "nodes"
      - Have each node have to contribute some resource!
  - "Proof of stake" is just another solution...
    - Which requires your money to be easy to steal!
    - Plus enshrines "he who has the gold, rules!"

- But there is an easier one: "Articulated Trust!"
  - Like the CAs: Use human-based agreements to agree on \( M \) trusted parties
    - Only \( \frac{1}{2} M + 1 \) need to actually be trustworthy!
The Irreversibility Problem

- A challenge: Buy $1500 worth of Bitcoin *now*, without:
  - Needing $1500 cash in hand, transferring money to an individual, or having a preexisting relationship with an exchange

- You *can't*!
  - Everything electronic in modern banking is by design reversible except for cryptocurrencies
    - This is designed for fraud mitigation: Ooops, something bad, undo undo...

- So the seller of a Bitcoin either must...
  - Take another irreversible payment ("Cash Only")
  - Have an established relationship so they can safely extend the buyer credit
  - Take a deposit from the buyer and wait a couple days
The Theft Problem...

- Irreversibility also makes things very easy to steal
  - Compromise the private key & that is all it takes!
  - See "How to make money with Bitcoin in 10 easy steps" by your's truly

- Result: **You can't store cryptocurrency on an Internet Connected Computer!**
  - The best host-based IDS is an unsecured Bitcoin wallet
  - So instead you have hardware devices, paper wallets, and other schemes intended to safeguard cryptocurrency
  - It is worse than money under the mattress:
    Stealing money under the mattress requires **physical access!**
The Decentralization Dream...

- "Trust Nobody"
  - The entire system is trustworthy but each actor is not
- Requires that there never be a small group that can change things...
- It is basically an article of faith that this is a good & necessary idea
  - But about the only thing it really buys is censorship-resistance
The Decentralization Reality

• Code is inevitably developed by only one or a few groups
  • And they can *and do* change it capriciously if it affects their money: When the Ethereum "DAO" theft occurred, the developers changed things to take *their* money back from the thief
  • Current debate to unlock another smart contract...

• Rewarded mining centralizes
  • Especially with ASICs and "Stealth ASICs" for proof of work mining
  • And the miners can *and do cheat*, such as enable "double spending" attacks against gambling sites

• Several just aren't decentralized at all
  • Trusted coordinator or seed nodes
  • Ability to override/freeze assets
The True Value of Cryptocurrencies: Censorship Resistance...

- There is (purportedly) no central authority to say "thou shalt not" or "thou shouldn't have"
  - Well, they exist but they don't care about your drug deals...
- If you believe there should be no central authorities...
  - Cryptocurrencies are the only solution for electronic payments
- But know this enables
  - Drug dealing, money laundering, crim2crim payments, gambling, attempts to hire hitmen etc...
  - Ease of theft of the cryptocurrencies themselves
  - Ransomware and extortion: estimates of half a billion dollars a year!
- And some minor "good" uses
  - Payments to Wikileaks and Backpage when they were under financial restrictions
Cryptocurrencies don't work unless you *need* censorship resistance

- *Any* volatile cryptocurrency transaction for real-world payments requires two currency conversion steps
  - It is the only way to remove the volatility risk
    - Which is why companies selling stuff aren't actually using Bitcoin, but a service that turns BTC into Actual Money™
    - And thanks to the irreversibility problem, buying is expensive
  - But if you believe in the cryptocurrency, you **must hodl!**
- Result is that the promised financial applications (cheap remittances etc) can *never apply* in volatile currencies like Bitcoin
  - Really Bitcoin et al are *only* appropriate for buying drugs, paying ransoms, hiring fake hitmen, money laundering...
  - Otherwise, use PayPal, Venmo, Zelle, MPasa, Square, etc etc etc...
Worse: Censorship Resistance Enables Crime

- Before the cybergangs had Liberty Reserve and still have Webmoney...
  - But Liberty Reserve got shut down by the feds (a shutdown that really screwed up the black market hackers), and WebMoney is Russia-only

- So the only censorship alternative is cash
  - Which requires mass ($1M ≅ 10 kg) and physical proximity

- So the cryptocurrencies are the only game in town!
  - The drug dealers hated Bitcoin in 2013, and hate them all still, but it is the only thing that works
  - Ransomware used to be Green Dot & Bitcoin, but Green Dot was forced to clean up its act
And "Stablecoins" are no better...

- Removing the two currency conversion steps requires **eliminating** volatility
- Building a stable cryptocurrency requires an entity to convert dollars to tokens and vice versa **at par**. AKA a "Bank" and "Banknotes"
  - Thus a centralized entity, so why bother with a "decentralized" blockchain? 😐
  - All other "algorithmic stablecoins" are snake oil that implode spectacularly
- There is now a choice for the bank
  - Either you become as regulated as PayPal & Visa
  - Or you have a "wildcat bank": This is banking in the 1800s
  - Or you have "Liberty Reserve" and the principals end up in jail
And The Big Stable-Coin, Tether, IS A FRAUD!!!

- Bitcoin's value is purely a speculative bubble
  - Somebody in the future will pay more than you paid today
- Bitcoin has a price equation based on supply/demand
  - New Bitcoin = (New $ + New Fake $s)
- Bubbles have been drive by fake $:
  - 2013: Willy-Bot on MtGox: Created fake $ in deposit in the Magic The Gathering Online Exchange Bitcoin exchange, bought Bitcoin
  - 2017: Tether: A stablecoin which unbanked Bitcoin exchanges use since they can't access the banking system. Roughly 1/3rd of the price runup then
  - 2020-21: Tether AGAIN: The Tether Printer go BRRRRR. Now in a situation where real new $ is deeply negative as they are adding billions of "dollars" a week in Tether to buy Bitcoin to back the Tether...
Practically Every Cryptocurrency is "Me Too" with some riff...

- There are lots of cryptocurrencies...
  - But in many ways they act the same: A public ledger structure and (perhaps) a purported decentralized nature
- Litecoin:
  - Bitcoin with a catchy slogan
- Dogecoin:
  - Bitcoin with a cool joke
- Ripple:
  - (Centralized) Bitcoin with an *unrelated* settlement structure
- IOTA:
  - (Centralized) Bitcoin but with trinary math 🤷‍♂️ and roll-thy-own cryptography 🤷‍♂️?!?!
- Monero:
  - Bitcoin with some better pseudonymity
- Zcash:
  - Bitcoin with *real* anonymity, err, "money laundering built in!"
- Ethereum:
  - Bitcoin with "smart contracts", unlicensed securities and million dollar bug bounties
Public Blockchain's Weak Security Guarantees

- "Public blockchains" protected by proof-of-whatever promise a "no central authorities" & "fully distributed trust" append-only data structure.

- But this isn't the case!

- Any lottery-based reward creates mining pools
  - Which means a few entities **can and do** control things: 5 entities effectively control Bitcoin with >50% of the hashrate

- The code developers also **can and do** act as central authorities
  - When ~10% of Ethereum was stolen from the "DAO", the developers rolled out a fork to undo the theft

- **And worse...**
Proof of Work's Economic Unsoundness

• Idea: The system wastes $x per hour to defend against potential attackers
• If an attacker needs to change the last $n$ hours of history...
  • They will need to spend at least $nx$, which acts as a floor
• This puts a ceiling on security as well: an attacker doesn't need to spend much more than $nx$
  • If an attacker can make more than $nx$ for an attack, they will!
• And its grossly inefficient:
  • The system is wasting $x$ per hour \textit{whether or not it is under attack}
• Oh, and there are services!
So The Security Must Be Either Weak or Inefficient

- Proof of work is provably wasteful
  - It *may* be possible to make "proof of stake" work, but that has different problems
- And there is no way to make proof of work cheap!
  - Proof of "whatever" protects up to the amount that "whatever" costs, *but not more!*
- So "articulated trust" is vastly cheaper
  - Take 10 trustworthy entities, each one has a Raspberry Pi that validates and signs transaction independently
    - In the end, 6 need to prove to be honest, but could easily process every Bitcoin transaction
  - This requires 100W of power and $500 worth of computers!, or 9 *orders of magnitude less power*
The Worm Problem....

- These cryptocurrencies form a closely connected peer-to-peer network
  - If you have an exploit that can compromise other nodes...
    You can make a self propagating attack (a "worm"), but do NOT DO SO
- Would be able to compromise **every node** in the P2P network in **seconds**
  - And you know that thing about "don't keep your cryptocurrency on an internet connected system"? Yeah, how many actually do that!
- Target a secondary cryptocurrency...
  - EG, Dogecoin is a fork of Luckycoin is a fork of Litecoin is a fork of Bitcoin....
  - With half a decade of **NO UPDATES!**
  - So search the post-fork Bitcoin code for indications of memory vulnerabilities
  - And write a worm that steals all the OTHER cryptocurrencies!
What About Non-Currency Blockchain Applications?

- Put A Bird Blockchain On It!
- "Private" or "Permissioned" Blockchain
  - Simply a cryptographically signed hashchain: Techniques known for 20+ years!
  - The only value gained is you say "Blockchain" and idiots respond with "Take My Money!"
- "Public" Blockchains are grossly inefficient and can't actually deliver on what they promise
- And those proposing "blockchain" don't actually understand the problem space!
A Concrete Example...

- A couple years ago there was a "Blockchain" class here at Berkeley
  - Yes, I screamed inside
  - I attended one session to give a short rebuttal...
    - But the two outside "experts" also present were delusional
- Concrete example: Vaccine supply chains...
  - You need to keep a vaccine supply chain suitably cold, if it gets too hot that is a problem...
  - One expert: "You can solve this in India with Blockchain!"
- BULLSHIT! You solve this with temperature-sensitive labels! At $1.50 each
- Proof of Nick's Iron Law of Blockchain: Blockchain solves exactly one problem: When someone says "you can solve X with Blockchain", they clearly don't know anything about X and should be ignored
But There Is One Innovative New Stupidity: "Smart Contracts"

- Idea! "Contracts are expensive!" 😐
  - So let's take standard things written in a formal language ("Legaleze")
  - And replace them with things written in a horrid language (that looks vaguely like JavaScript)
    - By default these "smart contracts" are fixed once released!
    - And this makes things cheaper how?

- And ditch the exception handling mechanism
  - If you can steal from a Smart Contract, are you actually violating the contract?
"Smart Contract" Reality: Public Finance-Bots

- They are really Public Finance-Bots
  - Small programs that perform money transfers
    - Finance bots are not new: The novelty is these finance bots are public and publicly accessible
  - Oh, and these aren't "distributed apps"

- Predictable Result: Million Dollar Bugs
  - The "DAO", a "voted distributed mutual fund as smart contract": Got ~10% of Ethereum before someone stole all the money!
  - The "Parity Multi-Signature Wallet" (an arrangement to add multiple-signature control to reduce theft probability)
  - The "Proof of Weak Hands 1.0" explicit Ponzi Scheme
The Rest Is Speedrunning
500 years of bad economics...

- Almost every cryptocurrency exchange is full of frauds banned in the 1930s
- Ponzi schemes without postal reply coupons, including explicit ponzies as "Smart Contracts"
- Tether, a "stablecoin" is almost certainly a wildcat bank from the 1800s
- Every tradable ICO is really an unregulated security just like the plagues in the South Sea Bubble of 1720
- Replicated rare tulips with rare cats on the Ethereum Blockchain as a "Smart Contract"! Time to party like it is 1637!
- And don't forget the goldbug-ism...
Smart Contracts and "Decentralized Finance": Speed Running the Speed Run

• "Hey, only Wall Street has previously benefitted from super-whiz-bangie techno innovations"
  • So lets instead build them as "Smart Contracts"?

• ONLY applications end up being:
  • Fraudulent stocks (ERC20 tokens)
  • Tulip Manias
  • Implicit ponzi schemes ("Yield Farming")
  • Explicit ponzi schemes
  • Front-running bots and fraudulent miners
  • And million dollar thefts seemingly on a near-daily basis
    • Not sure which is more, the thefts or the frauds ("Rugpulls")?
So The Space is Dismal

• The value is nonexistent
• The harms are great
• So avoid it...
• Or work on making it die in a fire