Ambush attacks on 160bit objectIDs & addresses

by Mysten Labs cryptography team *March 27, 2023*

Our recent internal audits and analysis re hash execution cost show that the **cost of 160bit hash collision attacks is in the feasibility range of just a few millions $, making it possible even for non-statewide adversaries.**

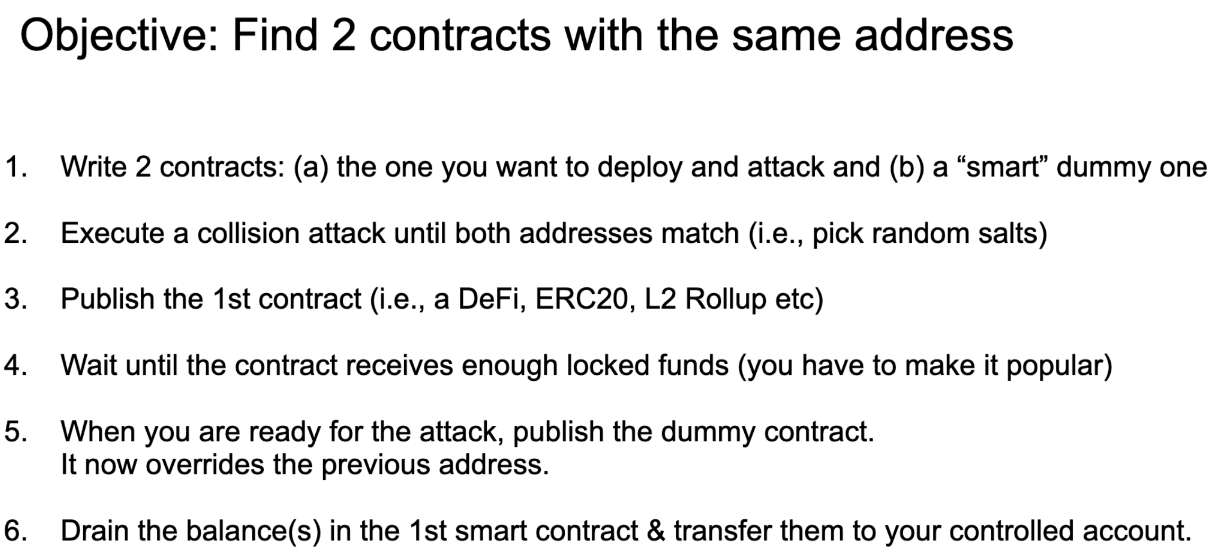
Proposal: switch to 256bit objectIDs (and inherently addresses too).

Impact and attacks

Before going into details on why this attack is feasible, let's assume someone can create two objectIDs that collide. This is an offline attack, no need to interact with the blockchain until you find a collision. This is possible because ObjectID creation is a deterministic function based on the transaction data or the parent ObjectID, depending on the object type. In any case, no unpredictable randomness is involved, and thus one could offline compute potential ObjectIDs.

Scenario 1 (ambush)

Offline compute two assets whose objectID collide; this can allow for ambush attacks (see our powerpoint presentation). Here is how a similar attack was possible on Ethereum (before geth client updates and related EIP-3607):



Objective: Find 2 contracts with the same address

1. Write 2 contracts: (a) the one you want to deploy and attack and (b) a “smart” dummy one
2. Execute a collision attack until both addresses match (i.e., pick random salts)
3. Publish the 1st contract (i.e., a DeFi, ERC20, L2 Rollup etc)
4. Wait until the contract receives enough locked funds (you have to make it popular)
5. When you are ready for the attack, publish the dummy contract.
   It now overrides the previous address.
6. Drain the balance(s) in the 1st smart contract & transfer them to your controlled account.

We can extrapolate the above Ethereum scenario to Sui. Note that in Sui where everything is an object such ambush scenarios can be applied at asset level (not contract-logic layer only). Ethereum is using 256bits for memory/object access, and they only use 160bits for addresses (including EOA - contract addresses), so the attack is more contained there.

Scenario 2 (live vs burnt objects)

Bridges and light clients typically work with objectIDs / txnIDs etc. It is technically possible to perform the above attack in such a way where the original object gets burnt, then the adversary recreates the same ID but for a different object. There are many business scenarios where this can be problematic and in theory it can even happen to fungible objects too, ie someone mints an $1B USDC object, then burns and creates another object with the same ID.

Scenario 3 (BFT safety)

Finding 2 transactions that result in the same ObjectID can break the safety of consensus protocols. Especially in the Fastpay protocol, these transactions can happen on parallel (single owner transactions). Note that we mostly focus on object collision as opposed to account address collision, because objects have states and any ID collision with another object is a BFT threat (for both ownership and asset feature reasons).

Mitigations

- Always check if the object exists before creating it → this is what Ethereum is doing (but for a while this issue was indeed exploitable as well).
- Always track burnt objects as well. Unfortunately, in a high tps chain like Sui, storing all historical objects to allow for lookups in each transaction would be prohibitive.
- Add randomness in the ObjectID creation (BFT block digests).
    - for shared objects: we can use the most recent Narwhal / Bullshark block hash (every couple of secs).
    - for single writer: we can use the hash of the previous epoch (every 24h).
    - unfortunately, our analysis shows that such collision attacks are in theory possible within a day (assuming someone had a portion of the bitcoin network power), which implies that adding randomness cannot provide a full countermeasure.
- Use 256bit objectIDs. This eventually what Sui adopted.

Note that the first 3 approaches only provide partial solutions, and especially with Fastpay, we would need protocol updates such as the following:

when a validator accepts a new txn it will:

- compute the ~100 possible object id that can be generated from txn, just by computing *h(epoch randomness* **|** *txn* **|** *i).*
- if any of them is already in its memory, abort the txn.
- we keep track of deleted ObjectID (in the current epoch).

- now, if someone tries to send *txn1, txn2* such that *h(epoch rand* | *txn1* | *i1)* = *h(epoch rand* | *txn2* | *i2)* for some *i1, i2*, honest validators will abort one of the transactions.

Summary: 256bit objects (and addresses) is a straightforward solution, and easier to implement than any other partial approach.

Educational content regarding security of hash functions

The 4 security properties of hash functions

There are 4 levels of security for a cryptographic hash function H:

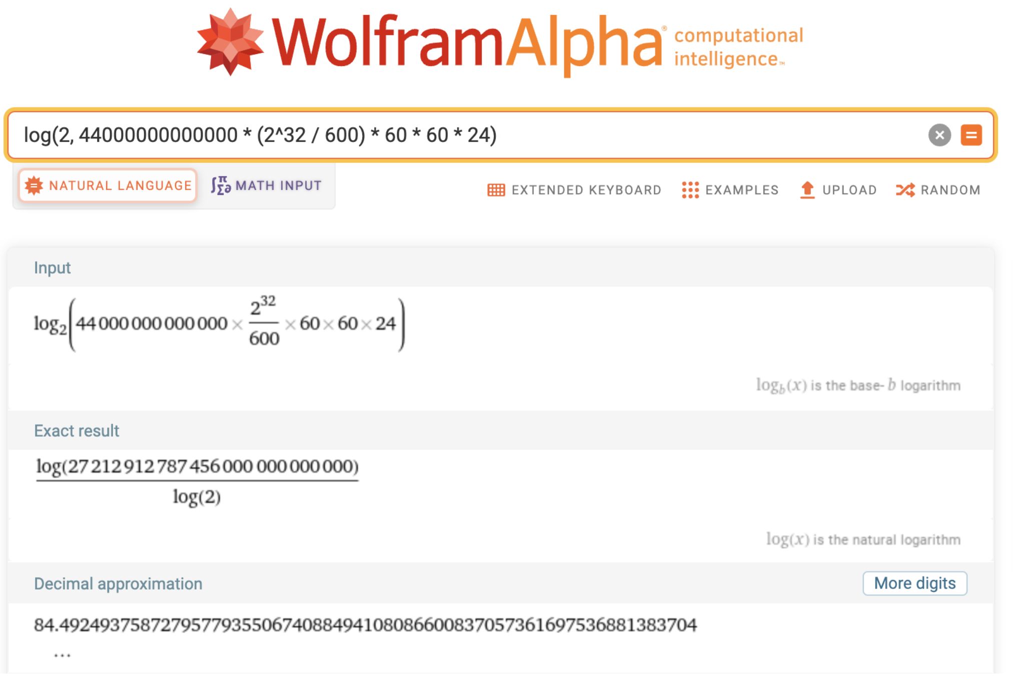
1. **Preimage resistance**: given a hash digest x, find a message m so that $H(m) = x$.
    - Security of modern secure hash functions against this attack is typically *2^n* where *n* = hash_digest_size.
    - For 160bit hashes, the effort is 2^160
    - For 256bit hashes, the effort is 2^256
2. **2nd-preimage resistance**: given $m1$, find another $m2 \neq m1$ so that $H(m1) = H(m2)$.
    - Security of modern secure hash functions against this attack is typically *2^n* where *n* = hash_digest_size.
    - For 160bit hashes, the effort is 2^160
    - For 256bit hashes, the effort is 2^256
4. **Multi-target 2nd-preimage resistance:** given a set of messages mX = {$m1$, $m2$ .. mK} find another mZ so that $H(m$Z$) = H($any of mX$)$.
    - The difference between this and 2nd-preimage is that now we target any element from a predefined set of messages, not just one $m1$.
    - Security of modern secure hash functions against this attack is typically *2^(n-k)* where *n* = hash_digest_size and k = log2(K) (where K = size of the set).

- As you see, **it's easier to find a multi-target collision attack vs preimage attacks**, because now you try to attack any existing element in a list, not just one.
- For 160bit hashes and 2^32 = 4 billion elements, the effort is 2^(160-32) = 2^128
- For 256bit hashes and 2^32 = 4 billion elements, the effort is 2^(256-32) = 2^224

5. **Collision resistance**: find any pair of $m1$ and $m2$ so that $H(m1)=H(m2)$.
    - Also known as the birthday paradox: If you survey a random group of just 23 people, there is actually about a 50–50 chance that two of them will have the same birthday. As you see it's not 1-out-of-365, because now we compare against many combinations of people-pairs, and we want at least one pair to match.
    - The difference between this and 2nd-preimage is that now we don't specifically target an m1, but we try to find any pair of elements (that we can even create offline) whose hashes collide.
    - Security of modern secure hash functions against this attack is typically *2^(n/2)* where *n* = hash_digest_size.
    - As you see, **it's by far easier to find a collision attack vs preimage attacks**, because now you try to find a random pair, not target a particular account/object etc.
    - **For 160bit hashes, the effort is 2^80**
    - For 256bit hashes, the effort is 2^128
        - Today, Bitcoin is operating at 2^84+ hashes per day, ~2^93 annually. Historically, this generally doubles every 1-2 years. The following calculation is converting difficulty target to hashes per day, as we see ~2^84.5

Cost of the 2^80 attack (this is the most important section)

The cost for the attack is estimated to be in the range of $1-10M. Reasoning:



log(2, 44000000000000 * (2^32 / 600) * 60 * 60 * 24)

Input

$$\log_2\left(44\,000\,000\,000\,000 \times \frac{2^{32}}{600} \times 60 \times 60 \times 24\right)$$

$\log_b(x)$ is the base- b logarithm

Exact result

$$\frac{\log(27\,212\,912\,787\,456\,000\,000\,000\,000)}{\log(2)}$$

$\log(x)$ is the natural logarithm

Decimal approximation

84.492493758727957793550674088494108086600837057361697536881383704
…

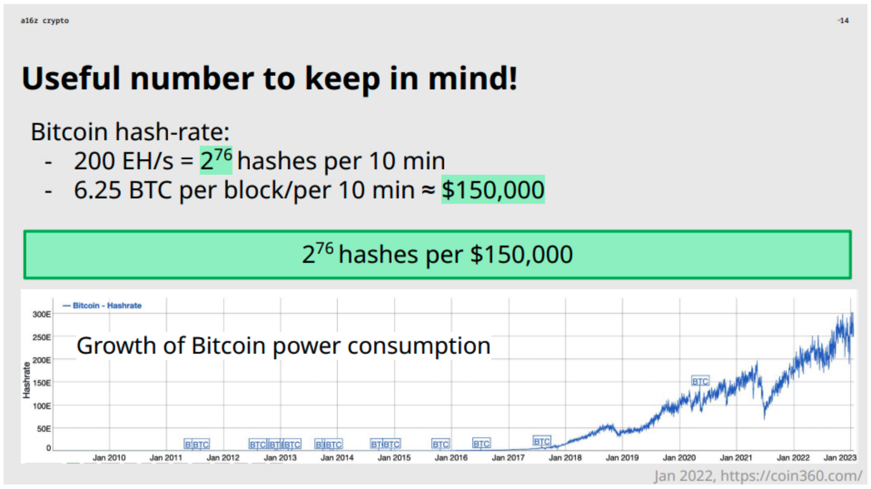
- The above implies that the Bitcoin network could in theory perform a 2^80 attack in about an hour!
- Based on the mining profit per day (which is a few million dollars), and taking into account that there is usually a profit margin of 5-50%, if we do the calculations the cost per hash can go down to about 1/(2^60) USD per hash invocation. Thus, for 2^80 we would need 2^20 = 1M USD.
- Typically mining is easier than collision attack generation because you need a slightly different setup for agents to communicate, but it's a straightforward infrastructure and we're aware of algorithms that search for collisions without requiring large storage requirements using techniques applied in rainbow tables. It would cost a bit more, but not necessarily significantly more.
- Note that back in 2017, it took researchers 2^65bits of effort and about $40K to break sha160. Since then, ASICS and GPUs and better access to cloud services

would allow for cheaper attack, reduced by a significant factor, thus it’s not a surprise that today a 2^80 attack would cost just a few millions.

- A similar analysis from FB research, showed that 2^92 collision attacks on AES cost about $4B, thus for 2^80 it would be about $1M; the results are consistent.

Notes

- Our claims have been discussed with various externals research teams and security experts in the industry, and the perceived feedback regarding recommended security levels for blockchain / contract / object address space is aligned with our claims. Our analysis is already getting some traction, even outside Sui, a16z has already created a slide around this:



a16z crypto 14

Useful number to keep in mind!

Bitcoin hash-rate:
- 200 EH/s = 2^{76} hashes per 10 min
- 6.25 BTC per block/per 10 min ≈ $150,000

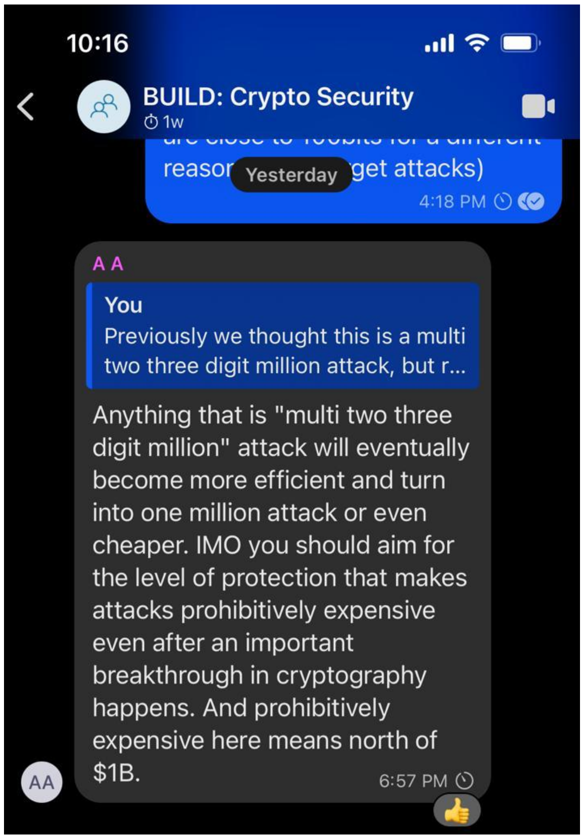
2^{76} hashes per $150,000

Growth of Bitcoin power consumption

Jan 2022, https://coin360.com/

Bitcoin - Hashrate
Hashrate
300E
250E
200E
150E
100E
50E
0
Jan 2010
Jan 2011
Jan 2012
Jan 2013
Jan 2014
Jan 2015
Jan 2016
Jan 2017
Jan 2018
Jan 2019
Jan 2020
Jan 2021
Jan 2022
Jan 2023
BTC

- Also based on discussions with security leads in the space, the general perception is that anything below $1B should be considered a threat, just to have room for cryptographic or CPU advancements. There was also a comment around the fact that even the “thought” of this being possible with a medium-scale adversary, it might disincentivize a few stable coin, bridges or asset providers to launch in the first place.


10:16

BUILD: Crypto Security

1w

reasor Yesterday get attacks)

4:18 PM

A A

You
Previously we thought this is a multi two three digit million attack, but r...

Anything that is "multi two three digit million" attack will eventually become more efficient and turn into one million attack or even cheaper. IMO you should aim for the level of protection that makes attacks prohibitively expensive even after an important breakthrough in cryptography happens. And prohibitively expensive here means north of $1B.

6:57 PM

- This analysis does not only apply to BFT applications, **160bit commitment schemes should also switch to 256bits**, ie the recent audit report to a16z over-collaterized sealed bid Ethereum auction contract here: https://github.com/a16z/auction-zoo/issues/2 (originally we thought it's a billion dollar cost attack, but this analysis proves it's in the single digit million range, especially because we don't deal with elliptic curve cryptography and it's just hash collisions on auction bidding prices).
- Note that adversaries can indirectly benefit from such attacks by on purpose causing reputation damage to the network and profit from price movements.
- We ethically informed other products and firms that we believed might be affected in the future if they stick with 20byte commitments (even outside Sui and the blockchain space) many months before this announcement.