Bitcoin Covenants

Opportunities and Challenges

Emin Gün Sirer
Ittay Eyal

Department of Computer Science
Cornell University

IC3 Initiative for Cryptocurrencies and Smart Contracts

State of Computer Security & How It Affects Us

- Our computing infrastructure is just not up to the task of holding high-value assets, like Bitcoin
- The default, low-energy state of an exchange is empty
  - From Bitcoinica to Mt. Gox to Bitfinex
- Clients lose their coins all the time
  - SFYL: “Sorry for your loss”
- Bitcoin has become a universal bug bounty

Possible Responses

- "Not our problem" is not a tenable strategy
  - Mobile and desktop OS vendors are woefully out of date
  - Web frameworks, databases, etc are even worse
  - We tried this strategy for years, and the resulting user experience is abysmal
- *We* can't fix world's security problems
- But we *can* provide better in-protocol mechanisms for securing coins

Vault Abstraction

- A secure place to hold your Bitcoins
- That enables you to recover your Bitcoins if they are stolen
- But does not impact fungibility
  - Regular payments are still final
- Takes away incentives for thieves to target Bitcoin nodes for theft

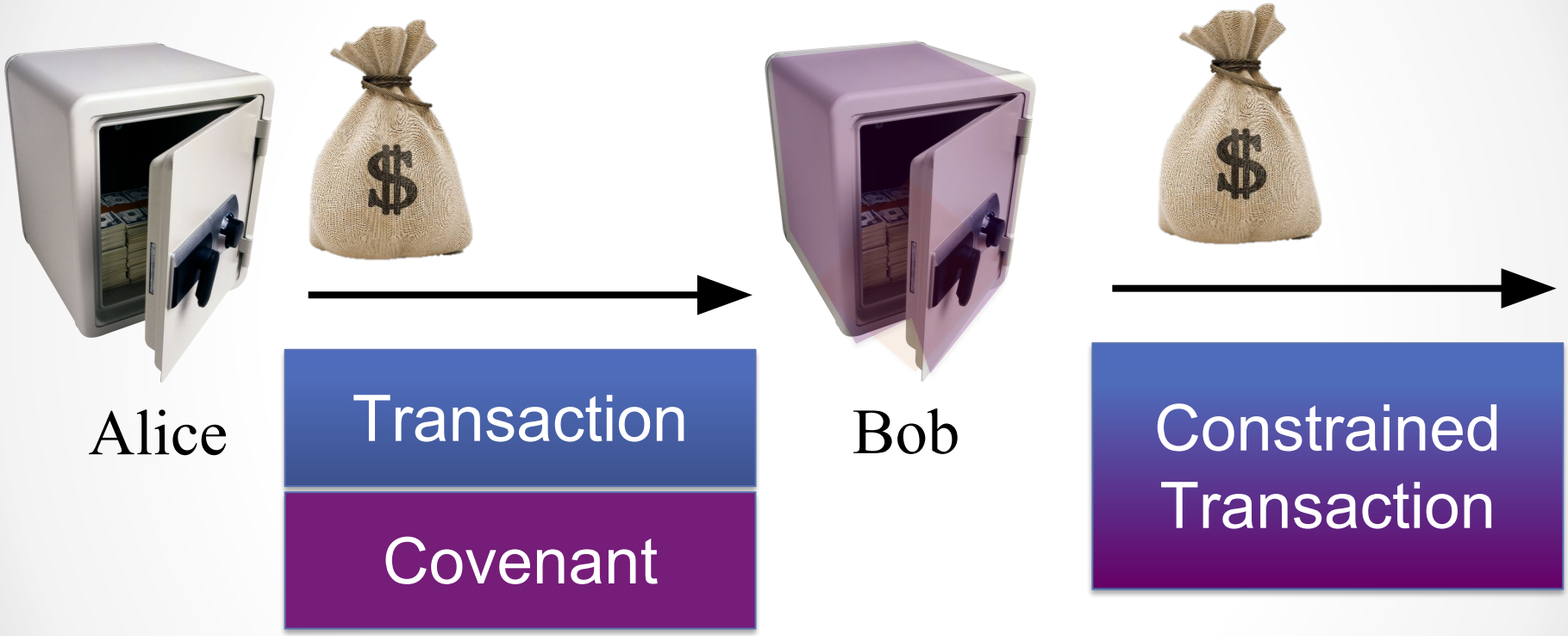
Vault Operation

- A vault has two keys:
  - Unvaulting Key
  - Recovery Key
- You place your coins in a vault, and specify an unvaulting period
- You can use the unvaulting key to move the coins to a regular address and spend as usual after the unvaulting period
- If a hacker gets your unvaulting key, you can use the recovery key to revert the transaction
- If a hacker gets your recovery key, you can burn the coins so they get nothing

Vault Implementation

- Many different ways of implementing vaults
- Special-casing vaults is not the right approach
- An elegant new abstraction, called Covenants, that requires only a single new opcode, enables vaults

Covenants



- A covenant is a restriction placed on the spend transaction

Covenants

- CheckOutputVerify
  - An output index, an amount, and a pattern
- Pattern is a sequence of bits, plus a mask to indicate which bits should be compared
  - Quines, aka self-reproducing scripts, are possible
- Bitcoin script used to be limited to data in the output and input scripts
  - CSequenceV and CLockTimeV already extended that
  - CheckOutputVerify adds a limited form of reflection

Covenants for Vaults

```
If
    <100> CheckSequenceVerify <keyDest> CheckSig
Else
    <100000> <patternVault> CheckOutputVerify <keyRecovery>  CheckSig
EndIf
```

- Vault Spend: a regular unvaulting needs to specify the unvaulting key, and specify a relative lock time with CSV
- Before the locktime expires, the funds can be moved to an output that retains the value and the covenant

Bonuses

Covenants for Colored Coins

- An incredibly popular use case
  - Gold
  - Precious metals
  - Real-estate
  - Vehicles
  - IoT
- All of the benefits of a grounded asset class, with the transferrability of Bitcoin
- Covenant ensures that coins that represent assets cannot be mixed with other assets

Covenants for Fraud Proofs

- Fraud proofs are a generally useful feature
- Trustless sidechains require fraud proofs to deter attacks on the reverse peg
- Easy to implement them with covenants

Covenants Concerns

- Complexity
  - Around 200 lines of code
- Fungibility
  - Covenants can be used to break fungibility
  - But wallets can detect non-fungible coins
  - Wallets can limit which covenants they accept
  - At the end of the day, fungibility is currently protected by the strongest means possible
- Overhead
  - Proportional to the size of the pattern to check
- Cost
  - Cost proportional to the size of the pattern

Early Deployment

- Elements Alpha implemented covenants
- Check for blog posts by Russel O'Connor

Summary

- CheckOutputVerify, a simple, elegant, single new opcode, enables us to make thefts a thing of the past, and opens up new use cases
  - No more SFYL
  - No more universal bug bounty

- Please send me your concerns in the next two weeks, and we'll collate them.