ECE 443/518 – Computer Cyber Security
Lecture 18 Bitcoin Management, Oblivious Transfer

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 23, 2024

Outline

Reading Assignment

- This lecture: Bitcoin Management, Oblivious Transfer
- Next lecture: Secure Multi-Party Computation

Outline

Bitcoin Management

Oblivious Transfer (OT)

Practical Considerations for Cryptocurrency

- Cryptocurrencies that are secure in theory are not necessarily so in practice.
  - Security: compromised hardware and software leak private keys.
  - Usability: complicated operations push people to look for practical solutions that are usually less secure.
  - Privacy: account owners can be targeted if identified physically.
- How to manage bitcoin to address these threats?
  - Weaknesses are constantly attacked because of the huge value associated with bitcoin.

Bitcoin Custody

- Will you trust someone to manage your bitcoin private keys?
- Yes: third-party custody
  - E.g. exchanges, banks, and brokerage firms.
  - Better usability: access bitcoin the save way as money.
  - No privacy: require physical identity.
  - Security concern: what if they cheat or are compromised?
  - Sometimes this is a must, e.g. to exchange between bitcoin and money, and to store bitcoin in retirement accounts.
- No: self-custody
  - Not your keys, not your coins.
  - Better security and privacy at the cost of usability.
  - Need a better understanding to achieve desired security and privacy, with a focus on managing private keys.
  - A third-party providing custody eventually relies on self-custody to manage bitcoin.

Bitcoin Transactions

- To send bitcoin
  1. Generate a bitcoin account as a public/private key pair.
  2. Obtain recipient's account address and create a transaction.
  3. Sign the transaction with the private key.
  4. Broadcast the signed transaction to the bitcoin network.
  5. Wait until the transaction to be included in the blockchain.
- To receive bitcoin
  1. Generate a bitcoin account as a public/private key pair.
  2. Let the other party know the account address and initiate the transaction as above.
  3. Wait until the transaction to be included in the blockchain.
- What are the threats associated with each step?
  - Clearly you would need to use computers for most of the steps.

Bitcoin Wallets

- Wallet: a hardware or software implemention of bitcoin protocol that one uses to interact with the bitcoin network.
  - Create accounts by generating public/private key pairs.
  - Access Internet to check account balances.
  - Access private keys to sign transactions.
  - Access Internet to broadcast signed transactions.
- Hot wallets: those holding private keys and being able to connect to Internet at the same time.
  - Easy to use and good for learning, e.g. wallet applications installed on your laptop and smartphones.
  - But the private key will leak if the wallet is compromised.
- Cold wallets: those holding private keys but without the capability of network communications.
  - Import transactions and export signed transactions through files that can be inspected to prevent potential leakage.
  - Should cold wallets support Bluetooth or USB connectivities?

Cold Wallet Security

- What if the cold wallet is compromised?
- Use password to control physical access.
  - Private keys are encrypted with password.
  - Counterintuitively, incorrect password should just give different private keys instead of any error message.
- Signed transactions can be validated without private keys so they are less likely to be affected.
  - Make sure the transaction has the correct recipient.
- What about generating private keys?
  - Compromised cold wallets may generate private keys that can be reproduced by adversaries.

Private Key Generation

- Bitcoin accounts are identified by ECDSA.
  - For simplicity, let's just write a bitcoin private key as $k_{pri} = a$ and the corresponding address as $k_{pub} = \alpha^a$.
- $k_{pri} = a$ has a length of 256 bits and should be generated from a true random number generator (TRNG).
- Could we use an online random number generator?
  - No, we should assume all such websites are compromised – adversaries will record all random number generated and watch for the corresponding bitcoin address.
- Cold wallets may take TRNG outputs manually to protect against attacks on key generation.
  - Rolling a die gives 1 out 6 possibilities. Rolling 100 dice will generate enough randomness for a 256-bit random number.
  - Clearly, you should not use a virtual dice roller from online.

Private Key Recovery

- What if the cold wallet (or other device for private key storage) is broken or lost?
  - Lost private keys cannot be recovered – all funds are lost.
- BIP-39 Mnemonic Code
  - Avoid human errors in handling bytes and binary strings.
  - A list of 2048 (2^{11}) easy-to-remember words.
  - Derive and reproduce a private key from 24 words.
  - Potentially with a password.
- Still, a reliable way to backup the words is required.
  - Clearly, you should not store them in your emails or any devices that connect to Internet.
  - Prevent lost of backups and leakage from backups – not a good idea to write them down on a piece of paper.
  - What about using multiple cold wallets?
  - Consider using multi-signature (multisig) for more effective use of multiple cold wallets.

Transactions and Privacy

- Privacy concern: multiple transactions on a single bitcoin account may reveal a lot of information about its owner.
- Ideally one account should be used twice.
  - Once for receiving bitcoin.
  - The other spends all by sending the remaining balance to a new account.
- Impact usability since the owner now need to generate, use, and backup multiple private keys.

Hierarchical Deterministic Wallets

- BIP-32: derive normal or hardened child keys from the parent key $k_{pri} = a$, each for a usable account.
- Normal child keys make it easier to derive accounts
  - For simplicty, consider ith child key as $k_{pri,i} = a + i$.
  - Then the ith account is $k_{pub,i} = \alpha^{a+i} = k_{pub} \times \alpha^i$.
  - The owner can derive the accounts without accessing the child keys or the parent key – less chance of leaking them.
- However, leaking a normal child key cause all normal child keys and the parent key to be leaked.
  - Use hardened child keys if that's a concern.
  - To derive accounts, the owner needs to access the hardened child keys and the parent key – more chance of leaking them.

Bitcoin (Full) Node

- The bitcoin network consists of bitcoin nodes that are connected as a peer-to-peer network.
  - Store and validate current blockchain.
  - Resolve fork by proof-of-work consensus.
  - Forward blockchain to other nodes.
- A bitcoin wallet needs to connect to a bitcoin node to check account balances and to broadcast signed transactions.
- Privacy concern: nodes may identify account owners by the IP addresses that their wallets use to connect to nodes.
  - Use of virtual private networks (VPNs) may hide IP addresses from nodes but will expose the same to the owner of VPNs.
- Run your own node and connect your wallets to it.
  - It is much more difficult to decide which node a signed transaction reaches first.

Outline

Bitcoin Management

Oblivious Transfer (OT)

Oblivious Transfer (OT)

- Alice runs a pay-per-view service that provides access to n messages $m_1, m_2, \ldots, m_n$.
- Bob would like to access a particular message m_k.
- Bob don't want to let Alice know what is k.
  - For privacy reasons.
- Bob don't want to pay Alice a lot of money to obtain all the messages in order to hide k.
- Let's consider the simple case for two messages ($n = 2$).
  - Alice's secret: m_1, m_2.
  - Bob's secret: $k \in \{1, 2\}$.
  - At the end, Bob learns m_k but not the other among the two messages, and Alice learns nothing about k.
- How could this even be possible?
  - Assume Alice and Bob are honest but curious.

Mechanism Design

- Alice's RSA key pair: $k_{pr} = (n = pq, d)$, $k_{pub} = (n, e)$.

1. Alice sends Bob two random messages x_1 and x_2.
2. Bob generates a random message y and sends Alice v.
   - $v = (y^e + x_k) \bmod n$.
3. Alice sends Bob m'_1 and m'_2.
   - $m'_1 = m_1 + ((v - x_1)^d \bmod n)$.
   - $m'_2 = m_2 + ((v - x_2)^d \bmod n)$.
4. Bob computes $m'_k - y$ to recover m_k.
   - For $k = 1$, RSA guarantees that $m'_1 = m_1 + ((v - x_1)^d \bmod n) = m_1 + (y^{ed} \bmod n) = m_1 + y$.
   - Same applies when $k = 2$.
   - So Bob indeed learns m_k.

Analysis for Alice

- The only piece of information Alice directly learns from Bob is the message v.
  - $v = (y^e + x_k) \bmod n$.
  - Note that Alice has no kwowledge about y and k.
- With x_1 and x_2, Alice may derive y_1 and y_2.
  - $y_1 = (v - x_1)^d \bmod n$.
  - $y_2 = (v - x_2)^d \bmod n$.
- $v \equiv y_1^e + x_1 \equiv y_2^e + x_2 \pmod{n}$.
  - Alice cannot decide which of y_1 and y_2 is y.
- Alice learns nothing about Bob's secret k.
  - No matter how powerful Alice is.

Analysis for Bob

- Assume $k = 1$ for Bob.
  - Bob will learn m_1.
  - Does Bob learn anything about m_2?
- Bob learns x_1, x_2, m'_1, m'_2 directly from Alice.
  - x_1 and x_2 are simply random messages, providing no information on m_2.
  - $m'_1 = m_1 + y$, having nothing to do with m_2.
- $m'_2 \equiv m_2 + (v - x_2)^d \equiv m_2 + (y^e + x_1 - x_2)^d \pmod{n}$.
  - Bob may learn m_2 if and only if he can decrypt the ciphertext $y^e + x_1 - x_2$ encrypted with Alice's public key.
  - Since Alice chooses x_1 and x_2, to decrypt $y^e + x_1 - x_2$ implies Bob could decrypt any message encrypted with Alice's public key – this breaks RSA.
- Bob, if computationally bounded, learns nothing about m_2.

Summary

- A lot of efforts to make bitcoin more usable in practice, improving its security and privacy.
- Oblivious transfer (OT) as a building block for more complicated protocols.