REMOTE SIDE-CHANNEL ATTACKS ON ANONYMOUS TRANSACTIONS
In Zcash & Monero

By

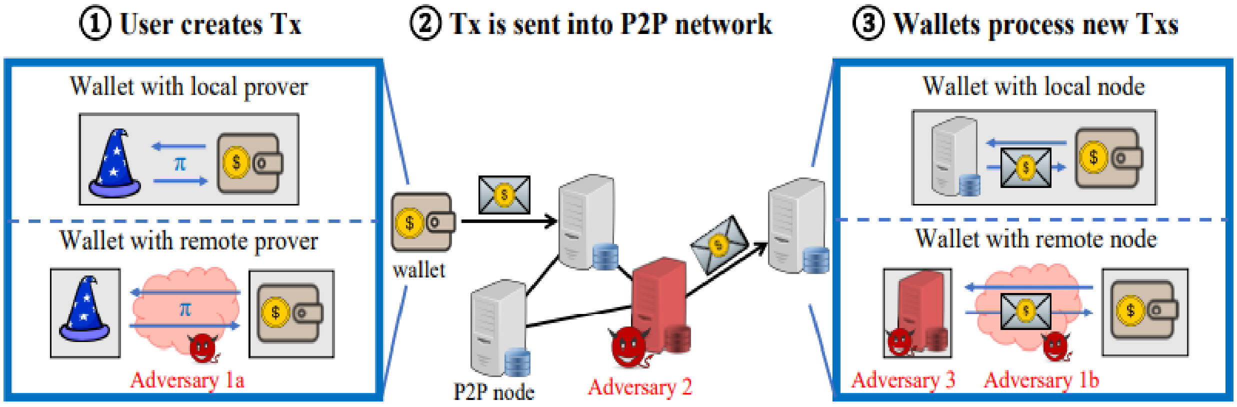
Ravi Sankar Gogineni

ABSTRACT

Privacy coins have outperformed recent trading, such as Zcash or Monero, that aim to provide cryptographic solid guarantees for transaction confidentiality and unlinkability. Although side-channel and traffic-analysis attacks let remote adversaries bypass these protections. These attacks enable an active remote adversary to identify the (secret) payee of any transaction that violates the privacy goals. Timing differences are large enough that the attacks can be mounted remotely over a WAN and link all the transactions by measuring the response time of that user's P2P node to certain requests. The attacks highlight the dangers of side-channel leakage in anonymous crypto-currencies, and the necessity to protect them against such attacks systematically.

INTRODUCTION

cryptocurrency is an encrypted data string that denotes a unit of currency. It is monitored and organized by a peer-to-peer network called a blockchain. Bitcoin, the largest cryptocurrency, is not private. Systems like Zcash, Monero, and several others offer privacy on a public blockchain and we focus on them as they are the two largest anonymous cryptocurrencies by market capitalization. Zcash and Monero use fairly advanced cryptographic primitives such as succinct zero-knowledge arguments (zkSNARKs) and ring signatures. we look at side channel information that is leaked by the implementation of different components in the system. Specifically, we look at timing side channels and traffic patterns, as measured by a remote network attacker. Any information leakage can invalidate the zero-knowledge property, and weaken or break the privacy guarantees of anonymous transactions. There are multiple attacks on transaction privacy in Zcash and Monero that exploit communication patterns or timing information leaked by different parts of the system. Taking a look at the systematic approach and the life cycle of an anonymous transaction as it traverses the system, we can identify the side-channel attacks and their impact on user privacy.



- The transaction is created in the payer's wallet, possibly with the help of a remote server to generate the necessary zero-knowledge proof to prove transaction validity.
- The transaction is transmitted through the P2P network.
- The transaction is received by the payee wallet, possibly with the help of a remote P2P node that records all transactions in the P2P network.

In Zcash, a user's wallet and P2P node are run in a single process. And for Monero, where wallets and nodes are run in separate processes, we show that receipt of a payment alters the communication pattern between a wallet and its node

CORE DESIGN CONCEPTS
PRIVACY-FOCUSED CRYPTOCURRENCIES SUCH AS ZCASH AND MONERO

These cryptocurrencies build on the UTXO model. Each user of the currency possesses one or more public keys and connects to a P2P network to send and receive transactions. The unspent transaction output is recorded in a blockchain.

PRIVACY GOALS

In UTXO the recipient produces a signature under a secret key. Currencies such as zcash and Monero aim to provide the following stronger privacy guarantees:

- confidentiality
- untraceability
- unlinkability
- user anonymity

PRIVACY TECHNIQUES

- Confidential transactions hide the amount of transacted funds, they only reveal a cryptographic commitment to the transacted amount.
- UTXO anonymity sets provide untraceability by concealing the identity of a transaction’s inputs.
- Obfuscated and diversified addresses guarantee unlikability. Diversified addresses enable a user to anonymously transact with multiple entities. From a single secret key users can create multiple public keys.
- Block chain scanning is a technical consequence of unlinkability. Users have to scan every new transaction and perform various cryptographic operations to check whether a transaction is intended for them.

SOFTWARE DEPLOYMENTS

We consider the following common deployment modes which refer to the interaction between a user’s wallet and a P2P node.

- Integrated
- Local
- Remote owned
- Remote third party

THE ANONYMOUS TRANSACTION LIFE CYCLE

To send a new transaction a user’s wallet selects some UTXOs and produces a zeroknowledge roof of validity for the transaction. The transaction is sent to the P2P node assigned to the wallet. P2P nodes store the transactions in the memory pool. P2P nodes share these transactions with connected wallets.

OVERVIEW OF THE ATTACKS

Threat Model: The attacks are remote side-channel attacks. We thus never assume that a victim's software is compromised. A network adversary passively monitors encrypted traffic between a victim's wallet and a remote service. A P2P adversary participates in the P2P network. The attacker may deviate from the P2P protocol. A remote node adversary controls a third-party P2P node and passively monitors the communication between a victim's wallet and node.

ATTACK TYPE 1

SIDE CHANNELS AT THE RECEIVING PARTY

The most practical and pervasive side channel attacks that we discovered affect the last stage of the anonymous transaction life cycle. These attacks enable remote adversaries to break the system's unlinkability and anonymity guarantees.

ATTACK GOALS

Our attacks target transaction unlinkability and user anonymity. The attacker's goals are to determine whether two transactions pay the same address and to determine how the user of a known address connects to the P2P network. We consider two different attack scenarios:

The adversary knows an anonymous public key sends a transaction to this key to determine which wallet the key owner uses to receive transactions. An honest user sends a transaction for which the adversary does not know the intended payee. In addition, both attack scenarios represent a break of user anonymity and can be bootstrapped for additional privacy violations.:

- IP address recovery
- Diversified address linkability
- Private key recovery

ATTACK STRATEGIES

Our attacks exploit a difference in the way that a wallet processes a transaction when it is the payee and when it is not. We develop two general attack strategies:

- Traffic analysis of wallet-to-node communication.
- Inferring wallet behavior from the P2P layer.

Both strategies apply not only when a transaction is created and sent into the P2P network, but also when it is included in a block.

ATTACK TYPE 2

SIDE CHANNELS AT THE SENDING PARTY

We initiate a study of attacks on the cryptographic tools that guarantee confidentiality and untraceability at transaction creation time – specifically succinct zero-knowledge arguments.

ATTACK GOALS

The transaction sender is responsible for ensuring confidentiality and untraceability as we argue below the most plausible target for a remote attack is to recover transaction amounts. Challenges Remote side-channel attacks on transaction creation face a number of challenges:

- Non-interactivity
- Ephemeral secrets
- High entropy secrets

ATTACK STRATEGY

We consider a cryptographic timing attack that exploits timing variations in arithmetic operations depending on the values of the operands such attacks have been studied for many cryptographic primitives.

ATTACKS ON UNLINKABILITY AND ANONYMITY IN ZCASH

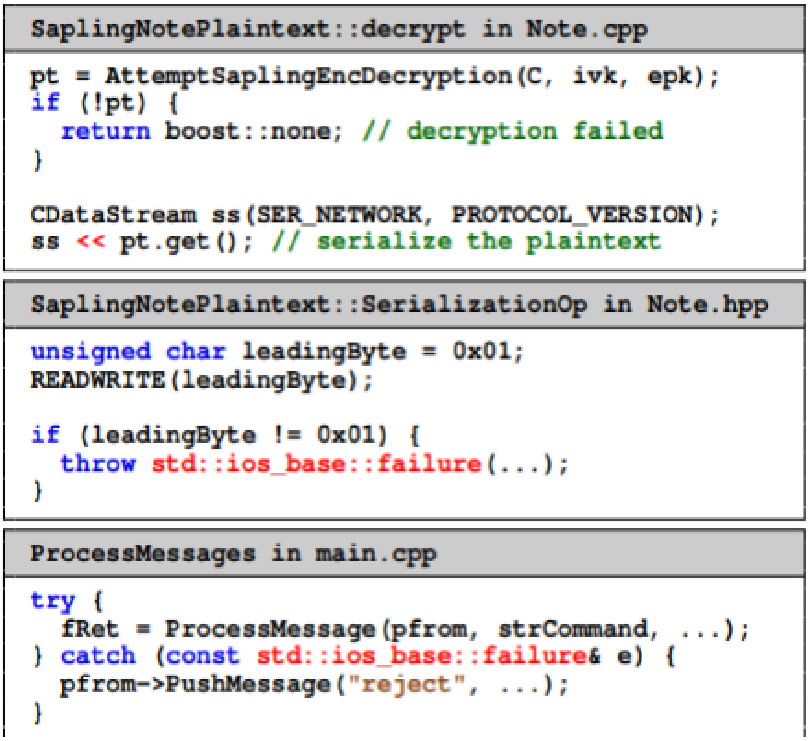
This attack exploits a lack of isolation between a user's wallet and the P2P node to leak wallet behaviors to a remote P2P adversary. In the Zcash client, the two components are part of a single process that sequentially processes received messages (including new transactions) and describe two side channel attacks that exploit this tight coupling.

- Zcash's diversified addresses are static Diffie-Hellman keys
- The private key is a scalar, ivk (the incoming viewing key). A diversified public key is of the form (Gd,PKd) where Gd is a random point in an elliptic curve group and PKd = ivk ·Gd
- Payment to the address (Gd,PKd) contains a UTXO (a Note commitment) of the form
- cm = Commit (Gd||PKd||v;rcm)

PING AND REJECT ATTACKS

- A (weak) form of "decryption oracle," which is used by both the PING and REJECT attacks, enables the adversary to determine if a particular ciphertext was correctly decoded by a node.
- A timing side-channel in transaction processing. If a Zcash wallet successfully decrypts a Note ciphertext, it checks that the opening of the Note commitment is valid (line 6 in Trial Decrypt)

- A timing side-channel in block processing. The above attack applies to unconfirmed transactions that enter a victim node’s memory pool. The same vulnerability also applies to the processing of transactions included in a mined block.
- Our initial attack, PING, take advantage of the close integration between the Zcash client's wallet and P2P components. More specifically, we make use of the Zcash client's serial processing of all incoming P2P messages, including those containing new transactions. Because of this, how quickly a transaction is processed affects how quickly a node processes other messages.
- REJECT takes use of a weakness in how some invalid transactions are handled. It enables a foe to make a transaction and get a "reject" response from the user's P2P node if they have the user's public key.
- Prior to the attack's patch. Top: The encrypted stream is serialized into a Note plaintext if decryption of a Note ciphertext C is successful. Middle: If the protocol version is not encoded in the first byte of the plaintext, an exception s raised. An exception is caught by the client's message-processing thread, which then sends a "reject" message to the peer that sent the improper transaction.



SaplingNotePlaintext::decrypt in Note.cpp

```
pt = AttemptSaplingEncDecryption(C, ivk, epk);
if (!pt) {
  return boost::none; // decryption failed
}

CDataStream ss(SER_NETWORK, PROTOCOL_VERSION);
ss << pt.get(); // serialize the plaintext
```

SaplingNotePlaintext::SerializationOp in Note.hpp

```
unsigned char leadingByte = 0x01;
READWRITE(leadingByte);

if (leadingByte != 0x01) {
  throw std::ios_base::failure(...);
}
```

ProcessMessages in main.cpp

```
try {
  fRet = ProcessMessage(pfrom, strCommand, ...);
} catch (const std::ios_base::failure& e) {
  pfrom->PushMessage("reject", ...);
}
```

ATTACKS BEYOND RECIPIENT DISCOVERY

When a transaction with a corrupted Note plaintext is included in a mined block, an odd side effect of the REJECT attack occurs: the transaction payee's client crashes when attempting to validate the block. This issue might result in a potent DoS attack vector if an attacker managed to obtain payment addresses for numerous Zcash users. Even worse, if the attacker is aware of the payment addresses of numerous Zcash miners, making such a DoS attack is used to reduce the network's mining power (for example, to eliminate mining competition or prepare for a 51 percent attack. Recovering keys using ECDH timing. A remote timing channel on Zcash's implementation of the ECDH key exchange, specifically the Elliptic curve multiplication ivk EPK in Trial Decrypt, is also revealed by the PING and REJECT attacks.

REMEDIATION

Running two Zcash nodes—a "firewall" node that connects to the P2P network and a local node that stores the user's keys but only communicates with the firewall—provides a straightforward defense against the kinds of attacks we demonstrate. Apart from the DoS attack, all our attacks are prevented by this configuration, which requires storing and validating the full blockchain twice. Given that the Zcash protocol is mostly non-interactive, side-channel resistance may have seemed like a minor issue. As demonstrated by our attacks, a single flaw in the in-band secret distribution process unintentionally permitted two-way communication between an attacker and a target, potentially exposing a remote time side channel on the Zcash non-interactive key-exchange mechanisms.

ATTACKS ON UNLINKABILITY AND ANONYMITY IN MONERO

UNLINKABILITY IN MONERO

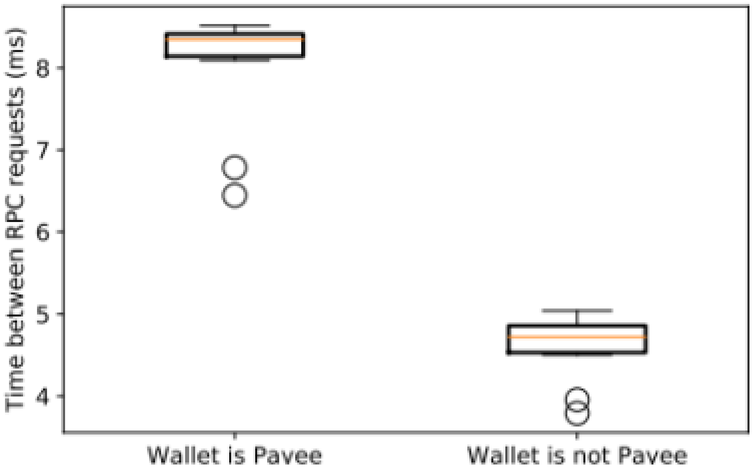
A high-level explanation of how stealth addresses, a method for generating a re-randomized public key for each transaction sent to the same recipient in order to ensure unlinkability, are used by Monero.

Monero user, Alice, has a public key of the form (A, B) = (aG, bG), where G is a base point belonging to the elliptic curve group. The pair of scalars (a,b) ∈ Z 2 q is the client's secret key. To receive funds from another user, the host, the client shares her public key (A, B) with the host. This key interchange is done by Diffie-Hellman key exchange.

Concretely, the host picks an ephemeral secret key r R←− Zq and computes P = H (rA)·G+B, client needs to prove knowledge of a scalar x such that P = xG. Given (P, R), she can compute this secret as P = H (rA)·G+B = (H (aR) +b | {z } x)·G

MONERO DEPLOYMENTS AND ATTACKS

Deployment types: Wallet and Remote nodes. Traffic Analysis Attacks for Remote Nodes, following an automated refresh, the wallet initially asks the node for a list of unconfirmed transactions and returns a list of hashes in return. Then, it asks the bodies for two different kinds of transactions: those that the wallet has never handled before and those that have already been viewed but in which the wallet is the payee.
Timing of Monero block requests. Plots the time between requests for blocks made by a wallet to a remote node, depending on whether the initial block contains a transaction for the wallet (left) or for another user (right). Twenty times the experiment is conducted
There will be Timing Attacks for Remote Nodes and Timing Attacks for Local Nodes.

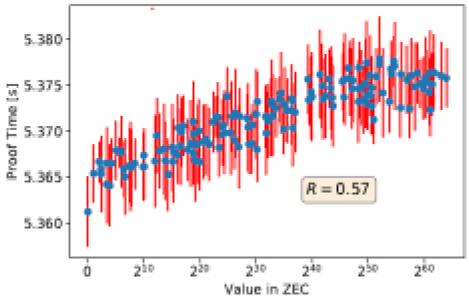


TIMING ATTACKS ON ZKSNARK PROVERS

The side channel attacks that we discussed in the previous sections exploited the flaws in the system design of P2P clients and wallets. Here we investigated the potential for side-channel vulnerabilities in succinct zero-knowledge arguments (zkSNARKs). In the below sections 6.1 and 6.2, it was demonstrated that timing attacks reveal information about transaction amounts in Zcash at the same time about the ineffectiveness of similar attacks for special-purpose proofs in Monero.

TIMING SIDE-CHANNELS IN THE ZCASH PROVER

Here we showed that in Zcash's zkSNARK system, proving times heavily depend on the value of the prover's witness. Particularly for anonymous transactions, proving times are heavily correlated with a transaction's confidential value. Zcash uses the Groth16 proof system. it is enough to know that the prover encodes the witness as a vector (a1,…., am) of field elements and that the prover's main computation form is: $\sum_{i=1}^{m} a_i G_i$,



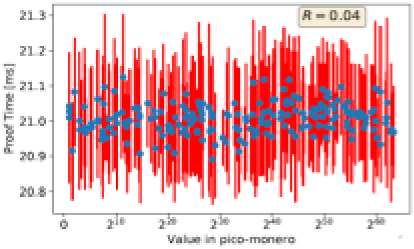
Correlation between transaction amount and prover time in Zcash

In order to evaluate the timing attack 200 transactions are picked out of the form 2^t for t uniformly random ([0,64)). 20 transactions were created for each of the 200 random amounts by randomizing over all the components. The left-most proof timings in the figure correspond to zero transaction amounts. Fingerprinting them is interesting to Zcash's dummy notes to obfuscate the number of UTXOs in a transaction. Users can create dummy UTXO's with zero value.

On comparing the attacks in the above sections, the above timing attack is not easy to apply. If a timing opportunity exists, we show that the resulting leakage allows the approximation of the private transaction amount. This attack serves as a warning about the dangers that may arise from non-constant-time cryptographic implementations. In the fig. proving time and transaction amount are strongly correlated (R = 0,57). we can also observe that the best-case and worst-case times differed by less than 20 ms (i.e less than 1% of the total prover time). Even though all proof takes a constant worst-case time, they will be still a small overhead.

ABSENCE OF TIMING SIDE-CHANNELS IN THE MONERO PROVER

In contrast to Zcash, Monero does not make use of a general purpose zk-SNARK system. Here the spender of the Monero transaction proves that the confidential transaction contains a value in the range of [0,2^64). A Crucial difference, when compared to Zcash, is that Bulletproofs not only operate in the binary decomposition of value but also on its bit-wise complement. The prover computes a Pedersen commitment of the form $\sum_{i=1}^{n} (a_L)_i \cdot G_i + (a_R)_i \cdot H_i$,



Correlation between transaction amount and prover time in Monero.

Similar to our Zcash experiment in Section 6.1, for a range of random transaction values, we timed 20 proofs with other witness elements chosen at random. We also observe that proof times are not constant, with variations of up to 0:5 milliseconds between proof times. The small resulting timing differences seem insufficient to reliably extract secret information from a single remote timing measurement.

RELATED WORK

In Monero, biases in the choice of anonymity set were shown to enable transaction tracing. In Zcash, the low volume of anonymous transactions was shown to enable the tracing of many transactions via usage pattern heuristics.

Our side-channel attacks complement a large body of work on the de-anonymization of cryptocurrency transactions. Many authors have shown that analyzing Bitcoin's public transaction graph breaks users' pseudonymity.

These attacks are not much effective on transactions with strong cryptographic anonymity guarantees, such as Zcash's fully shielded transactions. Whereas side-channel attacks exploit implementation flaws and bypass these protections to link or break the confidentiality of arbitrary transactions.

Our attacks further relate to the larger study of remote side channels in anonymization tools such as T or mix-networks.

CONCLUSION

A number of remote side-channel attacks on anonymous transaction systems such as Zcash and Monero are presented in the paper along with those powerful attacks on transaction unlinkability and user anonymity that exploit timing side-channels and communication patterns. Studied the impact of timing side channels on the zero-knowledge proof systems and showed Zcash's implementation leaks secret transaction data through the timing of proof generation. The attack discussed revealed a new facet in designing secure systems for anonymous transactions which will help in informing privacy-oriented crypto-currencies of the dangers of side-channel leakage and the results motivates the need for designing systems that proactively isolate user wallets from P2P interfaces also the development of constant-time implementations of cryptographic primitives.

REFERENCES

1. Alex Biryukov, Daniel Feher, and Giuseppe Vitto. Privacy aspects and subliminal channels in Zcash. In ACM SIGSAC Conference on Computer and Communications Security, 2019.
2. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. Technical report, Electric Coin Company, 2019. Version 2019.0.1 https://github.com/zcash/zips/ blob/d39ed0/protocol/protocol.pdf.
3. George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in Zcash. In 27th USENIX Security Symposium, pages 463–477, 2018.
4. Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In International Conference on Financial Cryptography and Data Security, pages 6–24. Springer, 2013.
5. Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Annual International Cryptology Conference, pages 104– 113. Springer, 1996.