FEDERAL RESERVE BANK of ST. LOUIS
CENTRAL TO AMERICA'S ECONOMY®

Tornado Cash and Blockchain Privacy: A Primer for Economists and Policymakers

Matthias Nadler and Fabian Schär

Abstract

This article explores non-custodial crypto asset mixers such as Tornado Cash. We analyze what types of mixers exist and how they work. We discuss opportunities and risks and offer an approach, based on voluntary disclosure, that would allow financial market regulators to combat money laundering and illicit activities, while allowing honest users to interact with privacy-enhancing protocols. We explain how crypto asset mixers play an important role on public blockchains and that privacy may be difficult to attain without them.

JEL codes: B27, D53, G18

Federal Reserve Bank of St. Louis *Review*, forthcoming.

1 INTRODUCTION

It is difficult to retain privacy on a public blockchain. In contrast to popular belief, permissionless blockchains are completely transparent. All confirmed transactions are publicly observable and stored as part of the blockchain's history. The users' identities are only protected through the use of addresses that act as pseudonyms. This setup allows public blockchains to operate without any intermediaries and creates a system where everyone can mathematically verify the legitimacy and integrity of transactions as well as the current state of the ledger; but the setup raises severe privacy concerns.

If someone obtains information that allows them to link a blockchain address to an entity, they may effectively observe that entity's entire transaction history and associated activity. Even if the entity uses multiple addresses, any link between these addresses may expose the fact that they belong to the same person. Moreover, the immutable and public nature of the data creates a setting where the data accrues over time and will always be available for analysis. The algorithms and tools to analyze the data will become more sophisticated, off-chain data more abundant, and computational constraints less relevant.

Fabian Schär is a professor for distributed ledger technologies and fintech at the University of Basel and the managing director of the Center for Innovative Finance at the Faculty of Business and Economics, University of Basel. Matthias Nadler is a PhD candidate at the Faculty of Business and Economics, University of Basel. The authors thank Tobias Bitterli, George Fortier, Andreas Glarner, Mitchell Goldberg, Emma Littlejohn, Remo Nyffenegger, Katrin Schuler, Dario Thürkauf, and the editors and referees of the *Review*.

© 2023, Federal Reserve Bank of St. Louis. The views expressed in this article are those of the author(s) and do not necessarily reflect the views of the Federal Reserve System, the Board of Governors, or the regional Federal Reserve Banks. Articles may be reprinted, reproduced, published, distributed, displayed, and transmitted in their entirety if copyright notice, author name(s), and full citation are included. Abstracts, synopses, and other derivative works may be made only with prior written permission of the Federal Reserve Bank of St. Louis.

To preserve some privacy, many users rely on so-called crypto asset mixers (sometimes also referred to as tumblers or privacy-enhancing protocols). Other ideas for achieving (partial) privacy on public blockchains exist,[1] but crypto asset mixers are currently the most widely used approach. Simply put, the goal of a crypto asset mixer is as follows: Various entities[2] deposit the same amount of a specific crypto asset to a mixer address. The mixer acts as a pool. Anyone who has contributed to the pool may then generate a new address and withdraw their funds without revealing the link between the depositor and withdrawal addresses. To be precise: Third parties can still observe the addresses that have deposited to and withdrawn from the pool, but given a large enough anonymity set (see Section 4.2), those third parties cannot link a *specific* depositor address to a *specific* withdrawal address. Thus, crypto asset mixers break the visible link between transactions.

To provide a simple example, let us assume that Alice has sent a crypto asset to Bob. Bob now knows Alice's public address and may potentially observe her account for other activity. To hide her future activity from Bob, Alice could deposit funds to a crypto asset mixer. In this mixer, the funds are pooled with deposits from Carl and Dave who have also deposited funds in equal denominations. When Alice later uses a different address to withdraw from the pool, Bob will not be able to tell whether the new account belongs to Alice, Carl, or Dave.

Given the high degree of transparency on public ledgers, there certainly is a legitimate privacy use case for crypto asset mixers. However, there is also strong evidence that crypto asset mixers are being used for money laundering and to hide traces of illicit activities. On various occasions, funds resulting from hacks have been deposited to Tornado Cash. Some of these hacks were allegedly conducted by the North Korean hacker group *Lazarus*. Estimates by crypto analytics firms suggest that almost 30% of the funds deposited to Tornado Cash have originated from illicit activities.[3] These circumstances have led to a lot of regulatory scrutiny.

On August 8, 2022, the U.S. Treasury's Office of Foreign Asset Control (OFAC) placed the Tornado Cash smart contracts on the Specially Designated Nationals and Blocked Persons (SDN) sanctions list, effectively making it illegal for U.S. citizens to interact with the Tornado Cash protocol. The OFAC has added custodial addresses (including custodial mixing services) to the SDN before, but this Tornado Cash sanction is the first time a non-custodial protocol has been targeted.

The goal of this article is to provide an interdisciplinary introduction to non-custodial crypto asset mixers, to create a foundation for economists and policymakers, and to enable further research at the intersection of privacy and illicit activity. We use Tornado Cash as an example to show how non-custodial crypto asset mixers work. We collect and present data that may be useful for researchers and policymakers, point toward new regulatory challenges, and present potential solutions to some of the problems.

Crypto assets in general is a highly interdisciplinary topic. The related topic of non-custodial crypto asset mixers is especially complex, and it is not possible to adequately discuss its challenges and opportunities without some technical background. Since this article is targeted mainly at economists and policymakers, we introduce some of the technical core concepts used in non-custodial crypto asset mixers in the next section. This allows readers without a technical background to understand how the protocol works and follow the analysis and discussion more easily.

[1] For example non-interactive stealth addresses, see https://github.com/ethereum/EIPs/blob/master/EIPS/eip-5564.md

[2] Note that it is also possible for the same entity to make several deposits. Either through the same or multiple depositor addresses.

[3] See https://blog.chainalysis.com/reports/tornado-cash-sanctions-challenges/

2 BACKGROUND

Crypto asset mixers can be implemented at various levels of technological sophistication. The simplest version is a custodial model, where a centralized service provider operates the mixing service. Users send their funds to the mixing service's public deposit address and specify a recipient address over a private channel. While this can work in principle, custodial mixers are entirely trust-based. First, users must trust the service provider to fulfill their obligation and transfer the funds to the deposit addresses. The service provider is in full control of the assets and could invest, lend, or potentially steal them. Second, since the centralized entity knows the link between the deposit and recipient address, the user's privacy depends on the service provider's ability and willingness to irrevocably dispose of the identifying data after the mixing operation has concluded.

Non-custodial crypto asset mixers do not require trust. Instead, they rely on cryptographic schemes that allow anyone to prove and independently verify the validity of a withdrawal without disclosing the link to a specific deposit. Moreover, the user does not have to share the identifying information with anyone and there is no liquidity risk, since the funds are locked and cannot be used in any other way. As such, non-custodial mixers can be created as an immutable and independent infrastructure, where no centralized entity can unilaterally control, alter, or delete the information.

This distinction is in line with FinCEN's guidance document[4] that differentiates between anonymizing service providers (money transmitters) and anonymizing software providers (not money transmitters). While custodial mixing services operate as money service providers and constitute a legal entity that can be regulated, the situation is less clear in the context of non-custodial mixing protocols. The FinCEN guidelines suggest that they do not fall within the domain of a money transmitter.

The basic challenge with non-custodial crypto asset mixers is that there are two conflicting goals. On the one hand, the protocol's purpose is to break the link between the depositor address and the recipient address. As such, it cannot store information that would allow anyone to see how deposits and withdrawals are linked. On the other hand, it must ensure that (i) withdrawals can be initiated only by entities who have previously deposited funds and (ii) any given deposit can be withdrawn only once.

In what follows we briefly lay the foundations and discuss the individual building blocks that are needed to allow for public validation of these two conditions without violating privacy. Readers who have a technical background and are familiar with smart contracts, hash functions, Merkle trees, and zkSNARKs can go directly to Section 3.

2.1 Smart Contracts

Most non-custodial crypto asset mixers (including Tornado Cash) are based on smart contracts. A smart contract is a script that is immutably stored on the blockchain and runs as part of the blockchain's consensus protocol.

The term smart contract was proposed by Szabo (1994, 1997). The first smart contract blockchain was proposed by Buterin (2014) and formalized by Wood (2015).

Smart contracts are accounts with their own distinct address. Instead of being controlled by a private key, smart contracts are controlled by code. By default, the contract logic cannot be changed after deployment. Any asset that is sent to the smart contract will remain in control of this smart contract until a condition in its code is met that will transfer the assets to a new address.

A smart contract has two main components: *functions* (or methods) and *state variables*.

Functions can be interpreted as the smart contract's action set. Any interaction with a smart contract is initiated by a transaction that specifies a function call. If Alice wants to interact with the smart contract, she issues a transaction. She sets the *recipient* address to the smart contract's address and adds

[4] See FIN-2019-G001, https://www.fincen.gov/sites/default/files/2019-05/FinCEN%20Guidance%20CVC%20FINAL%20508.pdf

information on what function she wants to call. If the function has arguments (parameters), she chooses these values and adds them to the transaction. As a simple example, imagine a smart contract that allows Alice to store a number. When she calls the `store()` function, she is expected to add a number of her choice as a function argument. When the transaction is confirmed, the smart contract function will be executed and the smart contract state updated. The update is reflected in the state variables.

State variables can be interpreted as the smart contract's long-term memory. Any information that must be stored by the smart contract will be assigned to a state variable. The values may change in line with the specific rules of the smart contract when a function is being executed. In our simple example, the value Alice chose as an argument for her function call would be stored in the contract state.

For a more detailed introduction to smart contracts and their applications in financial markets, see Schär (2021).

2.2 Hash Functions

Hash functions map an input of quasi-arbitrary length to a fixed-length output. Essentially, the output is a deterministic digest of the input data. More formally, let us define $H(x) = h$, where $H()$ is the hash function, x the input and h the resulting output, usually referred to as *hash value*. Hash functions have two main applications: checksums and cryptographic fingerprints.

Checksums are used to decrease the probability of undetected typos. Prominent examples include checksum applications in credit card and bank account numbers. To provide a simple example, let us assume that x is a basic account number without a checksum. Using x as an input for $H(x)$ would yield checksum h. The actual account number including the checksum can then be represented as $I = h||x$, where the two pipes represent a simple concatenation. A typo in I would likely create incompatible x and h values and allow anyone to detect the typo with probability $1 - \frac{1}{e}$, where e represents the size of the hash function's uniformly distributed mapping set. In other words, a two-digit checksum with $h \in \{0, \ldots, 99\}$ means that 99% of all typos can be detected through the validation of $h == H(x)$.

Cryptographic fingerprints are usually used to ensure inclusion and integrity of information or to set up schemes, in which the knowledge of x serves as a secret and h is stored as a reference for validation purposes. They therefore add the additional requirement of collision resistance. Collision resistance in the context of hash functions means that, given a function $H(x)$, it is infeasible to find more than one x for a given output h. This requires a very large e. Most cryptographic hash functions use $e \geq 2^{160}$. Moreover, the function must be one-way, referring to the property that it must be infeasible to invert the computation. In other words, one can verify that h is the digest of a given x but one cannot derive x from h.

A simple introduction to the application of cryptographic hash functions in the context of Bitcoin can be found in Berentsen and Schär (2018).

2.3 Merkle Trees

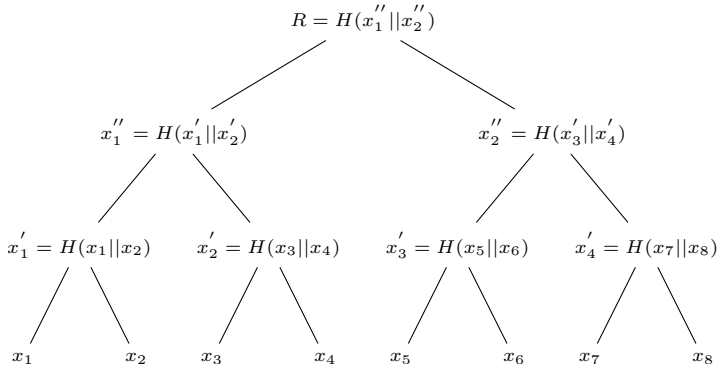
Let us assume that $\boldsymbol{x}$ is a vector of length N, with $\boldsymbol{x} = (x_1, \ldots, x_N)$. One could easily concatenate the elements of the vector and use $H(x_1||x_2||\ldots||x_{N-1}||x_N) = h$ to compute a hash value that acts as a fingerprint for $\boldsymbol{x}$. The problem with this approach is that h can be used to prove $x_i \in \boldsymbol{x}$ only if the prover has knowledge of the entire input vector $\boldsymbol{x}$. In other words, using the hash value to prove that some element x_i is part of the input vector $\boldsymbol{x}$ requires a large amount of input data and is unnecessarily inefficient. For these reasons, hash-based integrity proofs for multiple values usually rely on a data structure referred to as a *Merkle tree*.

Merkle trees hash the elements of an initial input vector pairwise.[5] Any x_i with an uneven index is hashed with the next element of the vector. In case of an uneven N, an element with value zero is

[5] The elements of the initial input vector are sometimes referred to as *leaves*.

Figure 1

Simple Example of a Merkle Tree with Eight Leaves



NOTE: In this simple Merkle tree, the example input vector $(x_1,\ldots,x_8)$ is hashed recursively and pairwise until a single hash, the root, remains.

appended. The resulting hash values are stored in a new vector $\boldsymbol{x}'$. The process is repeated with $\boldsymbol{x}'$ as the new input vector until the resulting output vector is of length 1. This value is referred to as the *Merkle root* and denoted by R. The number of rounds needed to arrive at the Merkle root is referred to as the height of the Merkle tree, with $log_2(N)$. In other words, a Merkle tree with an initial input vector of length 8 has height 3. This example is visualized in Figure 1.

To prove that x_i is part of $\boldsymbol{x}$ for a given R, one simply needs the values for x_i and $log_2(N)$ nodes along the path of the corresponding Merkle proof. Consider the following example: To construct a proof that x_5 is part of $\boldsymbol{x}$ for a given R, the values for x_6, x_4', and x_1'' would suffice. As such, one can construct very efficient inclusion proofs, particularly for initial input vectors with a large N.

2.4 zkSNARKs

zkSNARK stands for *zero knowledge, succinct, non-interactive argument of knowledge.* Let us discuss these properties one by one.

Argument of Knowledge: Intuitively this means that a person can construct a proof that demonstrates the knowledge of a secret value. The argument system is said to be *complete* if there are no false negatives and *sound* if the probability for false positives is negligible.

Zero Knowledge: The proof is said to be zero knowledge if it does not reveal the secret value or any other information besides the proof that a statement is true. In the context of Tornado Cash, Alice will be able to construct a proof that she has previously deposited to the Tornado Cash contract without having to reveal the specific deposit transaction.

Non-Interactive: The proof does not require any direct interaction between prover and verifier. In other words, a single message from the prover to the verifier is sufficient and the same proof can be used to convince any number of verifiers. This contrasts with interactive proofs, where each verifier sends multiple messages with cryptographic challenges to the prover. In a blockchain context, interactive argument systems are not feasible.

Succinct: The proof is referred to as succinct if it can be efficiently verified with respect to data size and verification runtime. Both properties are important in a blockchain context, as large input or storage data as well as complex on-chain computations would be infeasible.

Although zkSNARKS are an important part of Tornado Cash, a mathematical discussion is out of scope for this article. The reader who wishes to learn more about the mathematical principles behind zkSNARKS can turn to Petkus (2019), Thaler (2022) or Berentsen et al. (2022).

What we need to know for our analysis is that SNARKs use prover and verifier functions (more precisely, arithmetic circuits) that represent the argument system and are initialized through parameters, resulting from a trusted initiation (or setup) ceremony. We will treat the circuits as black boxes and assume that they satisfy the properties mentioned above. However, what is important to understand is that SNARK-based argument systems require an initiation ceremony in which some external entropy must be provided. These data must be discarded after the initiation ceremony, as anyone in possession of them could construct fake proofs.

Tornado Cash deals with this issue through a decentralized initiation ceremony in which anyone could sequentially contribute to the setup. If at least one contributor discards their secret contribution, it becomes impossible for anyone to generate fake proofs. In total, the Tornado Cash ceremony had 1,114 participants,[6] of which 664 remained anonymous and 450 provided their identity. Having both anonymous and identified participants is advantageous, as it mitigates collusion and blackmailing (potential problem with known actors) and sybil attacks where a single person participates with multiple anonymous accounts (potential problem with anonymous actors).

3 PROTOCOL DESCRIPTION

Tornado Cash is a smart contract-based crypto asset mixer that uses zkSNARKs to create a decentralized privacy-enhancing protocol. The code is open source and has been deployed on various blockchains, most notably Ethereum.

In this section we revisit our example from the introduction and analyze what is happening from a more technical perspective. We will use the protocol's two main functions and follow Alice as she makes a deposit and a privacy-enhancing withdrawal. The notation is mostly based on Khovratovich and Vladimirov (2019).

3.1 Deposits

Before depositing any funds, Alice chooses two random numbers, $k \in \mathbb{B}^{248}$ and $r \in \mathbb{B}^{248}$, with $\mathbb{B} = \{0,1\}$, where k is referred to as a nullifier and will later be used to prevent double spends, and r is a source of entropy. Note that both values are known only to Alice and must remain secret at all times. Alice uses a cryptographic hash function $H_1()$ to compute $C = H_1(k||r)$. C is usually referred to as a coin; however, it is easier to think of it as a cryptographic commitment that allows Alice to anonymously withdraw her funds at a later point in time, provided that she has securely stored k and r. If Alice loses access to this information, she will not be able to withdraw her funds. Similarly, if someone else learns the values for k and r, this person can steal Alice's funds.

After the local computations have concluded and Alice has properly stored her secret values, she issues a blockchain transaction, through which she calls the Tornado Cash contract's `deposit()` function. The transaction must include crypto assets in line with the pool's asset class and denomination. In other words, if Alice wants to deposit to the 1 ETH pool, she must add 1 ETH to the transaction. In addition, she includes data C, which will be used as a parameter in smart contract execution. Recall that C is a cryptographic hash value. Consequently, it represents Alice's choices of k and r through a cryptographic fingerprint, but it does not reveal the actual values of k and r.

When the transaction is executed and confirmed, the funds are locked in the Tornado Cash contract and the contract's state is updated accordingly. The contract stores a Merkle tree with height 20,[7] where

[6] See https://tornado-cash.medium.com/the-biggest-trusted-setup-ceremony-in-the-world-3c6ab9c8fffa

[7] This leads to 2^{20} = 1,048,576 leaves, which can be interpreted as an upper limit for the deposits in this pool.

each leaf can store a C from a deposit transaction. The C value from Alice's transaction is added to the next empty leaf and the Merkle proof values and root R are updated accordingly.[8]

Referring to our example in Figure 1, the new C value would be stored in x_i, where index i is the position of the first empty leaf in the tree.

3.2 Withdrawals

To withdraw, Alice has to pre-compute various data and issue a transaction that calls the `withdraw()` function of the Tornado Cash contract.

First, she creates a new Ethereum address A. This address will be the destination for her funds. Since it is a newly created address, there is no prior history that may reveal the link to Alice. Second, she computes a nullifier hash $\phi = H_2(k)$, where $H_2()$ is a different cryptographic hash function.

Third, she selects a recent R, which is a root that includes the C from her deposit and uses the prover circuit from the trusted setup ceremony to construct a proof P. The construction of this proof requires knowledge of k and r, as well as the leaf index i, which is the leaf position where her C is stored.

She then issues a blockchain transaction that calls the `withdraw()` function of the Tornado Cash contract, sending along A, R, P, and ϕ as arguments for the function.

As part of transaction execution, the contract verifies the following conditions:

1. Using the verifier circuit from the trusted setup ceremony, P must be a valid proof for the provided parameters A, R, and ϕ.
2. The presented nullifier hash ϕ has not been previously revealed as part of another withdraw transaction.

If both conditions can be verified successfully, the contract sends the corresponding value to A and adds ϕ to a list of previously revealed nullifier hashes.

Intuitively, condition 1 verifies if Alice has indeed deposited to the contract and thereby created a C. The Merkle root R is used as a summary for all C values. Using the Merkle tree structure and zkSNARK-based argument system, Alice can prove that she knows a tuple (k,r) that hashes to a C, which is part of the Merkle tree, without revealing which C she is referencing. The destination address A is included in the proof to make sure that it cannot be changed without invalidating the proof. Otherwise, anyone could use an already constructed proof, change the destination address, and try to front-run the transaction.

The second condition prevents double spends. Since the computation of the nullifier hash is also part of the prover circuit, any additional attempts to spend the same C more than once will result in a nullifier hash that has already been revealed. Similarly, setting the nullifier hash to an arbitrary value will create an invalid proof, since any $k' \neq k$ would lead to a different C value.

4 EMPIRICAL ANALYSIS

In this section we use on-chain data to give an overview of the usage and scale of Tornado Cash. First, we introduce the different asset pools and present monthly transaction counts and U.S. dollar volumes. We show the total scope of the protocol as well as its usage over time. Second, we analyze how protocol user anonymity can be quantified and how it evolved over time.

All the data in this section were gathered and verified with a personal Ethereum node.[9] The observation horizon is from each pool's first deposit up to and including block 15,449,617, the last block in

[8] The smart contract stores the last 100 values for R. The reason for this is that the withdrawal transaction must reference a specific R. If the contract allowed only for the latest R to be referenced, a dedicated attacker could front run withdrawals with deposits and thereby cause the withdrawals to fail.

[9] The exact software we used is a self-hosted Erigon archive node, see https://github.com/ledgerwatch/erigon

Table 1

Transaction Count and Usage Statistics of Tornado Cash

		Transactions				Deposit volume	
Pool	Observations (No. of days)	Deposit	Withdraw	Total	Total %	$	%
ETH 100	980	30,110	29,222	59,332	18.51	6,526,668,183	76.05
ETH 10	989	45,005	43,854	88,859	27.72	1,019,418,269	11.88
ETH 1	989	51,583	48,874	100,457	31.34	115,189,485	1.34
ETH 0.1	989	25,869	22,043	47,912	14.95	4,485,031	0.05
DAI 100,000	516	3,345	3,125	6,470	2.02	334,500,000	3.90
DAI 10,000	516	2,935	2,785	5,720	1.78	29,350,000	0.34
DAI 1,000	980	1,078	952	2,030	0.63	1,078,000	0.01
DAI 100	989	232	182	414	0.13	23,200	< 0.01
cDAI 5,000,000	509	114	114	228	0.07	12,204,941	0.14
cDAI 500,000	507	87	85	172	0.05	930,423	0.01
cDAI 50,000	503	118	117	235	0.07	126,033	< 0.01
cDai 5,000	980	5	3	8	< 0.01	520	< 0.01
USDT 10,000	859	4	4	8	< 0.01	40,000	< 0.01
USDT 1,000	963	1,165	876	2,041	0.64	1,165,000	0.01
USDT 100	980	354	281	635	0.20	35,400	< 0.01
USDC 1,000	960	572	501	1,073	0.33	572,000	0.01
USDC 100	980	150	111	261	0.08	15,000	< 0.01
wBTC 10	510	1,202	1,190	2,392	0.75	495,206,202	5.77
wBTC 1	517	1,033	1,030	2,063	0.64	40,247,218	0.47
wBTC 0.1	503	142	136	278	0.09	589,046	0.01

NOTE: Deposit volume based on transactions, converted using daily ETH/USD and BTC/USD rates on Binance at time of deposit. Data extend up to Block 15,449,617, built on August 31, 2022, end of day UTC.

SOURCE: Ethereum blockchain / On-Chain Analysis.

August 2022 UTC (approximately three weeks after the OFAC sanction). Furthermore, for any deposit to a Tornado Cash pool we consider the depositor to be the account that initiated the transaction that led to the deposit. For example, if the deposit was made from a multi-sig wallet, the account that initiated the multi-sig transaction will be considered the depositor.

4.1 Usage

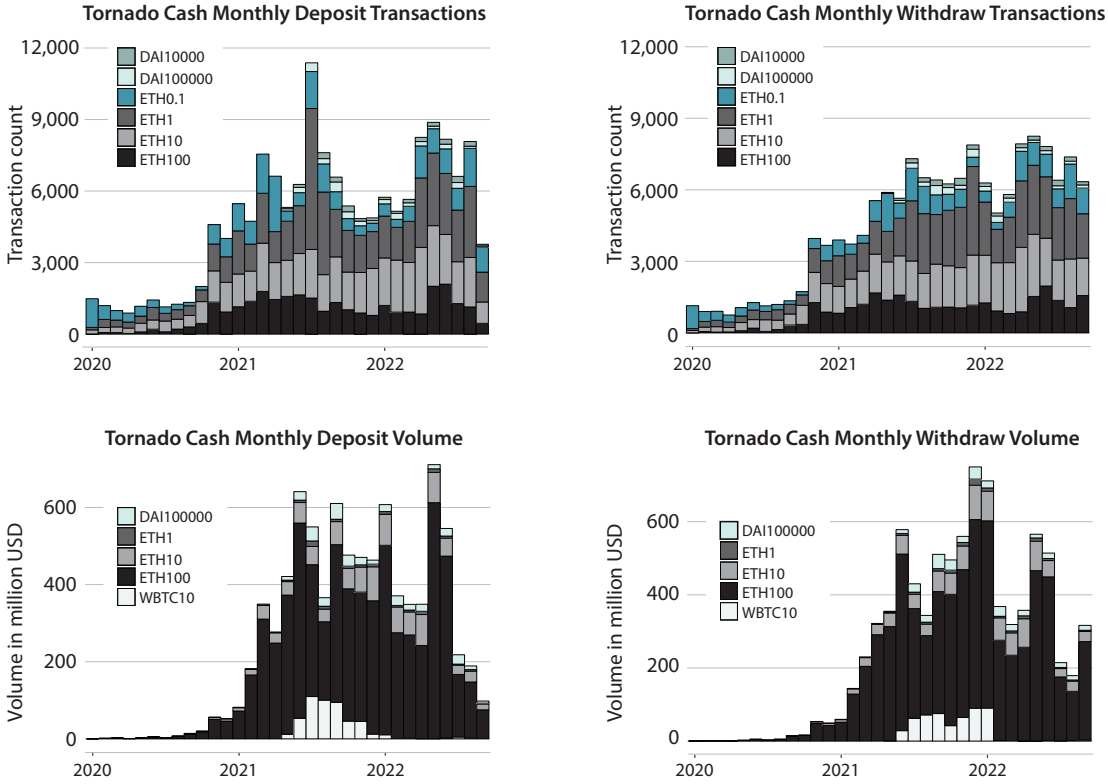
In total, assets equivalent to almost $8.6 billion, valued at the time of deposit, have been deposited to the various Tornado Cash pools in 164,975 transactions. Table 1 shows how the usage is split across different pools.

The ETH pools are by far used the most, by transaction count as well as volume. The ETH 100 pool processed the most by volume, but the ETH 10 and ETH 1 pools are used more frequently. Stablecoins account for less than 5% of the total volume, as shown in Table 1.

Figure 2 shows the monthly deposit/withdrawal volumes and the monthly deposit/withdrawal transaction counts for all pools with more than 1% of the total value.

Figure 2

Transaction Count and Volume by Pool



NOTE: Deposits and withdrawals of the most popular Tornado Cash pools by transaction count and volume.

SOURCE: Ethereum blockchain / On-Chain Analysis.

4.2 Anonymity Set

Funds that go through a mixer are not necessarily untraceable. Users can make mistakes, and there are various ways in which the depositor and recipient addresses may become linked, if a user does not interact with the protocol as intended.

Béres et al. (2020) and Wu et al. (2022) have proposed various approaches to de-anonymize Tornado Cash transactions, making use of address re-use, transaction activity, transaction cost choice, or more complex transaction graph and network analyses. For a discussion on how to de-anonymize Tornado Cash transactions or an extended discussion on the anonymity set, refer to the above papers.

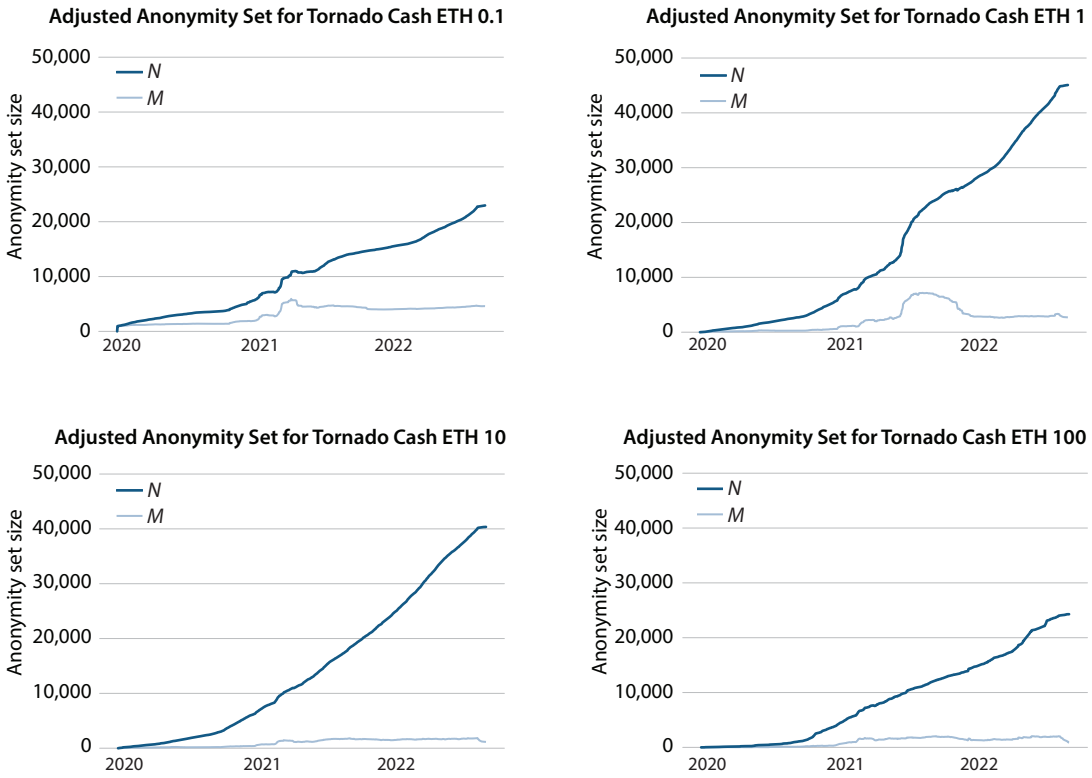
For our purposes here, we use a conservative approach and only link addresses, where the link can be established with certainty, i.e., cases where the same address is used for deposits and withdrawals. Similar to Wu et al. (2022), we find a significant number of actors that use the same address to deposit and withdraw their funds. While this seems surprising at first, there are various reasons why this may be the case, including but not limited to careless use of the protocol, testing transactions, or external incentive mechanisms.[10]

Figure 3 shows the adjusted anonymity set. At any point we know that there are M outstanding withdrawals that may belong to any one of N deposit addresses. If a person withdraws anonymously, this will lead to a decrease of M. N, on the other hand, is affected only if the link between deposit and

[10] See https://tornado-cash.medium.com/tornado-cash-governance-proposal-a55c5c7d0703, especially the section "Anonymity Mining"

Figure 3

Deposit and Withdrawal—Adjusted Anonymity Set



NOTE: *M* is the number of outstanding withdrawals from the respective Tornado Cash ETH pool. *N* is the adjusted anonymity set: the number of deposit transactions that could be linked to a withdrawal transaction.

SOURCE: Ethereum blockchain / On-Chain Analysis.

withdrawal is publicly revealed, i.e., if the person does not preserve their anonymity. If *M* reaches 0 at any point, *N* will also be reset, as it becomes apparent that none of the previous deposits are still in the pool. At any point in time, *M* and *N* can be interpreted as follows: There are *M* outstanding deposits that belong to a subset of the anonymity set, which has *N* elements.

Additionally, Khovratovich and Vladimirov (2019) have pointed out that the reference to the Merkle roots may also have negative effects on the anonymity set. Suppose that there are currently 10 deposits and 5 withdrawals. Under normal circumstances, the 5 withdrawals could be from any depositor. However, if all withdrawals use a root value from a root state R_j, with $j \leq 5$, we know that all of the withdrawals link back to the initial 5 deposits.

5 DISCUSSION

In this section we discuss two important aspects of non-custodial crypto asset mixers. First, we analyze different types of privacy and suggest an approach how Tornado Cash might be usable in a regulated environment and allow honest users to benefit from partial privacy. Second, we discuss some regulatory challenges—namely, third-party tainting and redeployments.

5.1 Partial Privacy Proposal

The fundamental problem with any discussion around privacy is that there are good arguments for both sides. On the one hand, privacy-enhancing tools are being used by criminals and this is certainly an undesirable outcome. On the other hand, privacy may be desirable. For example, it may serve as an insurance against excessive centralization of power and contribute toward the resilience of a democratic system. An optimal solution will likely lie somewhere between perfect privacy and perfect observability. Ideally, the infrastructure would generate a separating equilibrium between honest and dishonest actors and allow the honest ones to remain partially private.

In the following section, we argue that Tornado-like privacy-enhancing tools might be able to generate such a separating equilibrium. We then introduce a conceptual framework to differentiate between various types of privacy and discuss potential outcomes of various policy responses.

Separating Equilibrium

Tornado Cash can break the observable link between a deposit and a withdrawal address, but it cannot hide the information that someone has received funds from a Tornado Cash pool. These data are clearly visible on-chain and can be collected and analyzed by anyone. Consequently, if Alice has received funds from a Tornado Cash address, any potential counterparty can see this information on the blockchain and decide if they want to interact with Alice.

Recall that there are two distinct reasons why Alice may have chosen to use Tornado Cash. In the first case, let's say Alice did indeed commit a crime and is using Tornado Cash to hide the fact that the funds originated from an illicit activity. In the second case, let's say the funds came from a legitimate source and Alice is using Tornado Cash to retain some privacy on the blockchain.

Regulation may want to differentiate between these two cases. We argue that there is a relatively straightforward way to generate a separating equilibrium. If Alice's funds come from a legitimate source, she can easily share a cryptographic proof that links her deposit to her withdrawal address.[11] She can choose with whom she would like this proof to be shared, allowing the counterparty to analyze the blockchain as if she had never used Tornado Cash. If for example, Alice wants to deposit funds with her bank, she could provide a cryptographic proof to the bank that allows them to analyze the transaction graph as if Alice had never used Tornado Cash. At the same time, Alice does not have to reveal her linked transaction history to the entire network.

In contrast, the bad-acting version of Alice—let's call her Malice—will not be able to provide this proof, as the disclosure of the link would reveal the illicit on-chain origin of Malice's funds to her counterparty.

Regulated financial intermediaries will only accept the funds if the customer is willing and able to provide proof of the funds' origins. Similarly, merchants who sell a good or service above a legal threshold value are legally obliged to file these transactions and have strong incentives to ask for a proof of origin. Otherwise, they might be in violation of the law and face challenges when trying to use the funds for which they cannot provide information about the origin.

This process creates a situation where honest actors can remain partially anonymous, while dishonest actors will face severe *search and matching* cost to find a counterparty that is willing to take the risk and accept the funds without the proof of origin. In fact, this is similar to how cash transactions are handled today. If someone wants to deposit cash with financial intermediaries or use large amounts of cash to pay for goods or services, they have to provide a proof of origin for the funds. Blockchain-based non-custodial crypto asset mixers allow for an easier and more reliable proof than cash-based transactions.

[11] The standard frontend for Tornado Cash has provided a compliance tool, allowing the user to generate a report, including the cryptographic proof.

Table 2

Privacy Assessment by Regulatory Scheme

		Counterparty privacy	
	Public chain privacy	Backward-looking	Forward-looking
No mixing protocols	—	—	—
Mixing with partial disclosure	✓	—	✓*
Mixing with no disclosure	✓	✓	✓

NOTE: A comparison of privacy properties under various policy responses to non-custodial crypto asset mixers.

*Conditional on another ex-post privacy-enhancing event.

Note, that the use of a non-custodial crypto asset mixer is not free. Any smart contract interaction is subject to transaction fees. Moreover, the switch to a new address may require new token approvals (see ERC-20 token standard[12]), which are also subject to a transaction fee. While it is important to be aware of the existence of these costs, they can be mitigated, depending on the scalability properties of the blockchain or a switch to a layer 2, i.e., a scalability solution built on top of the blockchain network.

Forward- vs. Backward-Looking Privacy

A regulatory regime built around voluntary disclosure would allow Alice to use privacy-enhancing protocols by sharing some information with her specific counterparty. For obvious reasons, this voluntary disclosure scheme is not a perfect privacy solution. However, it allows honest actors to retain partial privacy and to choose with whom they want to share (parts of) their transaction graph.

To understand the exact implications of this approach, it might be useful to differentiate between *forward-looking privacy* and *backward-looking privacy*. We define the terms as follows: A privacy-enhancing event satisfies forward-looking privacy if it does not allow the observer to link a known transaction that occurred before the privacy-enhancing event to future transactions that happen after the privacy-enhancing event. Similarly, a privacy-enhancing event satisfies backward-looking privacy if it does not allow the observer to link a known transaction that occurred after the privacy-enhancing event to past transactions that happened before the privacy-enhancing event.

The voluntary disclosure scheme is very limited with respect to backward-looking privacy. If Alice is asked to share proof that allows the counterparty to link her current address to the pre-mixing address, that counterparty would be able to look back and scrutinize Alice's transaction history. For forward-looking privacy, however, Alice can simply run her funds through a new privacy-enhancing event. If her interactions with the counterparty are concluded, there is no reason to disclose the new proof to that specific counterparty. Thus, forward-looking privacy can be established.

Table 2 shows the privacy implications of various policy decisions. In a world without any privacy-enhancing protocols, the entire transaction history would be visible on the blockchain. Anyone could observe every single transaction. Privacy would be nonexistent.

In a world with no restrictions on the use of privacy-enhancing protocols, it would be possible to achieve public chain privacy and both forms of counterparty privacy, i.e., forward- and backward-looking. However, it would not be possible for a counterparty to observe if they are accepting funds that have previously been used in illicit activities.

A scheme that is based on privacy-enhancing protocols and voluntary disclosure would allow public chain privacy and forward-looking privacy, with respect to the specific counterparty.

[12] A standard interface for fungible tokens on Ethereum, see https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

Practical Examples

***Example 1*:** Let us assume that Bob has withdrawn crypto assets from custody with a regulated financial intermediary and stores these assets on his non-custodial wallet. He decides to subsequently use a non-custodial crypto asset mixer to break the publicly observable link between the custodian and his current address. Neither the custodian nor anyone else will be able to link the Tornado Cash deposit to the Tornado Cash withdrawal. He then uses his new address to buy something from merchant Alice. Alice observes the blockchain and can easily see that the funds originate from a Tornado Cash pool. She will ask for a cryptographic proof that allows her to link the Tornado Cash withdrawal to a deposit and observes that the funds originate from a legitimate source. Alice stores this proof.

To ensure that Bob cannot observe her future transactions, Alice decides to use Tornado Cash herself, thereby establishing forward-looking privacy against Bob. When she later wants to use the funds, her counterparty will ask for the proof of her Tornado Cash interaction, as well as the proof of the previous Tornado Cash interaction, allowing the counterparty to trace the payment back to the point when it was last held by a centralized entity.

***Example 2*:** Let us assume that Alice uses 10 distinct non-custodial addresses for various on-chain activities. She wants to consolidate these addresses and deposit the funds with a custodian. In the absence of a crypto asset mixer, anyone could easily see that the funds from these 10 accounts are sent to the same address. Anyone, particularly those who have interacted with one of Alice's addresses, could easily come to the conclusion that the other 9 addresses must also belong to Alice and analyze every single one of her past transactions across multiple accounts. A non-custodial crypto asset mixer would allow Alice to break the link between these accounts and share the information only with the party who needs to know about the funds' origin, i.e., the custodian.

5.2 Regulatory Challenges

Third-Party Tainting

Funds deposited in a non-custodial crypto asset mixer can be withdrawn to any address and, critically, there is no way to reject or block such a transaction if the recipient does not wish to receive mixed funds. Receiving funds from a sanctioned entity is a criminal offense. A malicious actor could therefore send their funds to a sanctioned, non-custodial crypto asset mixer and withdraw them to addresses that are publicly associated with another entity. For an observer, there is no way to determine whether the receiving party interacted with the mixer or not; and the receiving party will not be able to prove they were not involved. Recall that, while it is possible to demonstrate you interacted with the mixer via the cryptographic proof, it is impossible to show that you are not in possession of such a proof. The burden is placed on the receiving party who must act and try to resolve the situation. One potential way of doing so is to provably dispose of the tainted assets in a way that is publicly observable. In a blockchain context, this means to send an amount equal to the received funds to a known and accepted "burn address"—an address where no one controls the private key. The measure is not perfect, as it would still place a burden in the form of a mandatory action and transaction fees on the receiving party, but at least it would not allow a third party to freeze someone else's address or generate legal trouble. The OFAC has acknowledged this form of attack and addressed it on November 8, 2022, in an addendum to the Tornado Cash sanctions, stating that they will not prioritize enforcement regarding these "dusting" transactions.[13]

Further questions arise in any smart contract protocol that acts as an open marketplace with peer-to-peer sales or auctions. The problem here is that the seller has no control over the origin of the assets they receive and no information about the identity of the buyer. To mitigate this, the marketplace would have to restrict the participants and only allow users with a confirmed identity.

[13] See https://home.treasury.gov/policy-issues/financial-sanctions/faqs/1078

Redeployment

One of the goals of sanctions is to elicit a change of behavior from the sanctioned entity. Smart contract-based protocols without privileged access are immutable and therefore not capable of changing their behavior by design. Sanctions in this context are more akin to a ban of the protocol.

Smart contracts are deployed and stored in the form of code on the blockchain. The code for each smart contract is public and can be read directly from the blockchain. This makes it trivial for anyone to copy and redeploy a new instance of any protocol at a different address. As a result, regulatory actions against a specific smart contract address are at best a temporary solution. Redeployments with identical code will force frequent updates of a sanctions list and lead to a game of "whack-a-mole."

Slight variations in the code can make the situation even more challenging, as it may be unclear if something is a functional copy of a sanctioned protocol or a new implementation that must be treated and analyzed separately.

In contrast, regulating on- and off-ramps would have the advantage that it places the burden of proof on the individual who wants to transfer the funds to the financial intermediary.

6 CONCLUSION

Privacy-enhancing protocols are a very interesting innovation, with pros and cons. On the one hand, privacy may be desirable and is not necessarily associated with illicit activity. On the other hand, there is strong evidence that privacy-enhancing protocols such as Tornado Cash are also used by malicious actors and may potentially allow them to cover their tracks.

The policy action set is bounded by the two extreme choices: (i) not allowing any interactions with privacy-enhancing protocols and (ii) accepting all funds from privacy-enhancing protocols, with no questions asked. We argue that neither of these extreme responses constitutes a good strategy. There are various reasons why someone chooses to interact with a privacy-enhancing protocol. Any extreme choice ignores the existence of these different use cases and creates an inefficient outcome. An optimal solution will likely lie somewhere between perfect privacy and perfect observability.

Consequently, we suggest an approach that could potentially create a separating equilibrium that allows honest participants to achieve partial privacy. Moreover, we propose a distinction between different types of privacy, namely, public chain privacy as well as forward-looking and backward-looking counterparty privacy.

It is important to highlight the fact that a user of privacy-enhancing protocols can voluntarily share information with any third party. This information can contain a cryptographic proof that reveals how their withdrawal transaction is linked to a prior deposit. The entity that receives this information can verify the proof mathematically and analyze the user's transaction graph as if they had never used the privacy-enhancing protocol. Yet, the user does not have to share this information publicly, allowing them to preserve public chain privacy.

Financial intermediaries (or merchants in case of a large sale) may ask for this proof. The approach essentially treats the privacy-enhancing protocol as an independent protocol and regulates the on- and off-ramps. This is similar to how cash transactions are regulated, with the big difference that cash does not involve an immutable transaction history. As such, a blockchain proof is much more reliable than any proof that involves cash.

We conclude that non-custodial crypto asset mixers are an interesting innovation and demonstrate the power of zero knowledge proofs. They provide honest users with the option not to share their transaction history publicly and use public blockchains similarly to other electronic payment systems.

Yet, the risks are real and should also not be underestimated. Some level of regulation is necessary and perfect privacy will not be possible in a regulated environment. It is crucial to regulate on- and off-ramps (including centralized on-chain protocols) and enforce AML (anti-money laundering) and CFT (combating the financing of terrorism) regulations through these intermediaries. If this is done properly, crypto asset mixers such as Tornado Cash may become an integral part of public blockchain infrastructure.

REFERENCES

Berentsen, Aleksander and Schär, Fabian. "A Short Introduction to the World of Cryptocurrencies." Federal Reserve Bank of St. Louis *Review*, First Quarter 2018, pp. 1-16; https://doi.org/10.20955/r.2018.1-16.

Berentsen, Aleksander et al. "A Walk-through of a Simple Zk-STARK Proof." 2022; http://dx.doi.org/10.2139/ssrn.4308637.

Béres, Ferenc et al. "Blockchain Is Watching You: Profiling and Deanonymizing Ethereum Users." 2020; https://arxiv.org/abs/2005.14051.

Buterin, Vitalik. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." 2014; https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

Khovratovich, Dmitry and Vladimirov, Mikhail. "Tornado Privacy Solution Cryptographic Review Version 1.1." 2019; retrieved August 15, 2022, before being taken offline; https://web.archive.org/web/20220808144457/https://tornado.cash/audits/TornadoCash_cryptographic_review_ABDK.pdf.

Petkus, Maksym. "Why and How zk-SNARK Works: Definitive Explanation." 2019; https://arxiv.org/abs/1906.07221.

Schär, Fabian. "Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets." Federal Reserve Bank of St. Louis *Review*, Second Quarter 2021, pp. 153-74; https://doi.org/10.20955/r.103.153-74.

Thaler, Justin. "Proofs, Arguments, and Zero Knowledge." 2022; https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf.

Szabo, Nick. "Smart Contracts." 1994; http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.

Szabo, Nick. "The Idea of Smart Contracts." 1997; https://nakamotoinstitute.org/the-idea-of-smart-contracts/.

Wood, Gavin. "Ethereum: A Secure Decentralised Generalised Transaction Ledger." 2015; https://ethereum.github.io/yellowpaper/paper.pdf.

Wu, Mike et al. "Tutela: An Open-Source Tool for Assessing User-Privacy on Ethereum and Tornado Cash." 2022; https://arxiv.org/abs/2201.06811.