DOI:10.1145/3132696

When it comes to anonymizing cryptocurrencies, one size most definitely does not fit all.

BY DANIEL GENKIN, DIMITRIOS PAPADOPOULOS, AND CHARALAMPOS PAPAMANTHOU

Privacy in Decentralized Cryptocurrencies

CRYPTOCURRENCIES PROMISE TO revolutionize the financial industry, forever changing the way we transfer money. Instead of relying on a central authority (for example, a government entity or a bank) to issue and manage money, cryptocurrencies rely on the mathematical design and security proofs of the underlying cryptographic protocols. Using cryptography and distributed algorithms, cryptocurrencies offer a fully decentralized setting where no single entity can monitor or block the transfer of funds. Cryptocurrencies have grown from early prototypes to a global phenomenon with millions of participating individuals and institutions.[17] Bitcoin[28] was the first such currency launched in 2009 and in the years since has grown to a market capitalization of over $15 billion (as of January 2017). This has led to the emergence of many alternative cryptocurrencies with additional services or different properties as well as to a fruitful line of academic research.

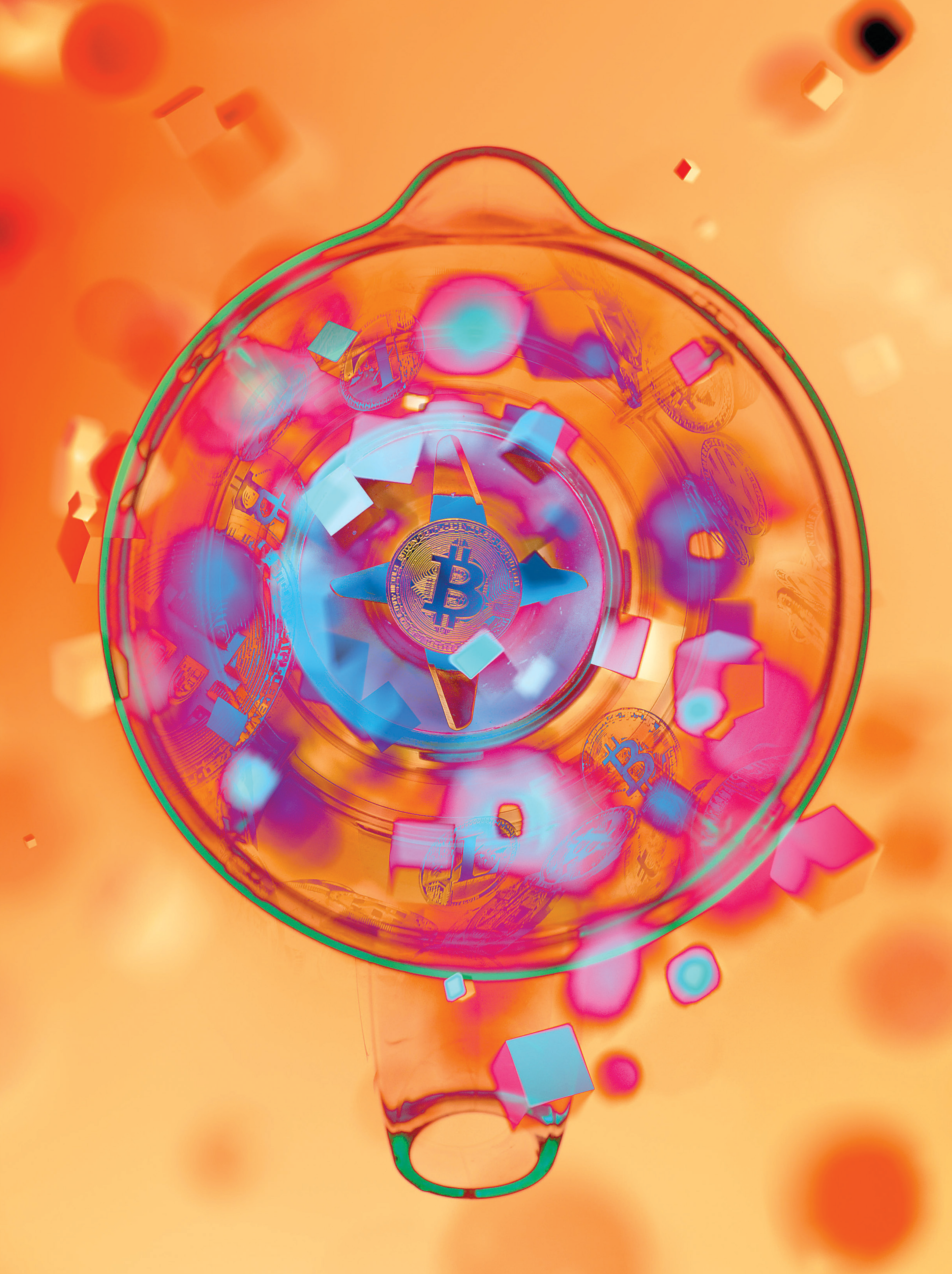
Apart from its other benefits (decentralized architecture, small transaction fees, among others), Bitcoin's design attempts to provide some level of "pseudonymity" by not directly publishing the identities of the participating parities. Every user interacts with the network by establishing a public address that acts as a "pseudonymous identity." In practice, there is no bound on the number of addresses a user can create; therefore there exists no single address a user can be related with. However, this pseudonymity is far from the desired *unlinkability* property in centralized e-cash protocols,[11] where when Alice sends an amount to Bob, the original source of these funds cannot be deduced. The reason for this problem is that in most decentralized cryptocurrencies all transaction information (payer and payee address, amount, among others) is publicly visible, stored in a distributed data structure called *blockchain* (for example, see www.blockchain.info). Therefore, an attacker can easily observe how money flows. This can lead to quite devastating deanomyization attacks and therefore there is a need for cryptocurrencies with stronger privacy guarantees.

In this article, we review widely studied mechanisms for achieving privacy in blockchain-based cryptocurrencies such as Bitcoin. We focus on mixing services that can be used as a privacy overlay on top of a cryptocurrency; and privacy-preserving alternative coins that,

» key insights

- **While blockchain-based cryptocurrencies like Bitcoin do not directly reveal users' identities, they are often prone to deanonymization attacks. By observing the flow of transactions stored in the public blockchain, third parties can make accurate guesses about the identities of involved individuals.**
- **Existing privacy-enhancing techniques for cryptocurrencies mostly come in two flavors: Mixing overlay protocols that can be executed on top of an existing cryptocurrency to hide the flow of funds among a set of participants, and alternative privacy-preserving cryptocurrencies that use advanced cryptographic techniques to achieve strong user privacy by design.**
- **We review and compare solutions from both techniques.**

IMAGE BY ANDRIJ BORYS ASSOCIATES/SHUTTERSTOCK



by design, aim to achieve strong privacy properties. We discuss and compare the privacy guarantees achieved by known mechanisms, as well as their performance and practical adoption.

Background: Bitcoin and Privacy

Bitcoin in a nutshell. The full mechanics of the Bitcoin protocol are rather involved and we refer interested readers to Bonneau et al.[7] and Nakamoto[28] (also see the "Inside Risks" column p. 20 in this issue). In the sequel, we provide a high-level abstraction of the protocol, highlighting the aspects that have the most impact on user privacy. A Bitcoin user participates in the protocol by first generating a cryptographic public/private key pair. The first operates as her public address: she can use it to send money to or receive money from other users in the same way one uses a bank account. Unlike a bank account though, a user can generate as many public/private key pairs as she wants—even one for every transaction. A simple transaction from user A to B contains a declaration of "A sends x bitcoins to B" signed with A's secret key.

These transactions are propagated via a flooding mechanism over an ad hoc, peer-to-peer network and are thus visible to every participant. Special users known as *miners* collect transactions and store them into blocks. These blocks are subsequently stored in a global public ledger of transactions known as blockchain. This chain is a sequential order of blocks, each of which references the previous one. Who appends the latest block is decided in a randomized manner using a proof-of-work mechanism that generally guarantees that the amount of blocks a miner gets to generate (receiving a corresponding miner's fee) is proportional to the ratio of its computational power over the total power of all miners in the protocol.

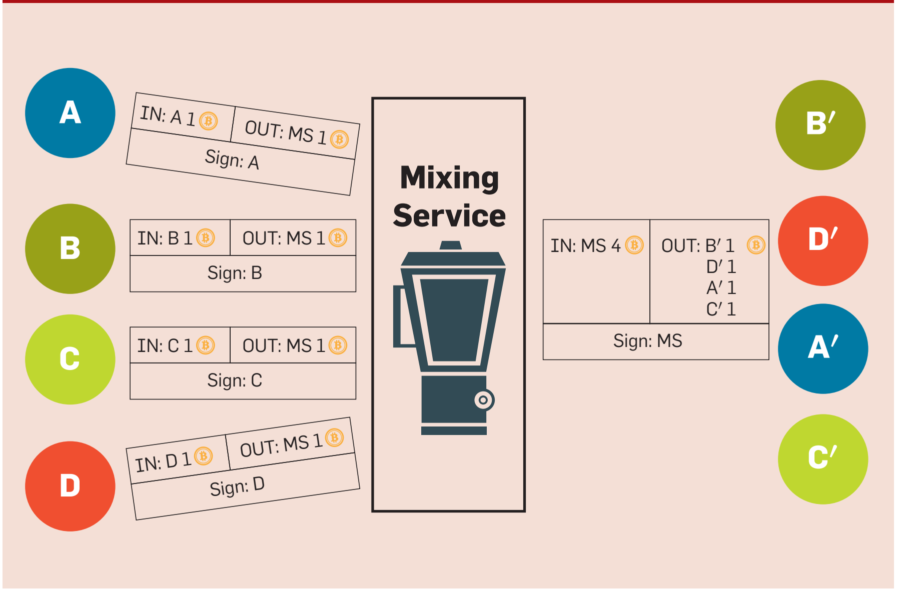
In order for a miner to add a transaction in the next block it must first be validated. Consider the case of the transaction from A to B. Before adding it to the current block, the miner must check that it is signed by A, and A did not previously spend these bitcoins. The former is easy to achieve given the public key of A (embedded in the transaction). The latter can be verified by tracing A's entire transaction history to check that the bitcoins in question where not previously spent (in practice, it suffices to just trace unspent transactions).

Deanonymization attacks. This scenario highlights a crucial issue regarding Bitcoin's privacy: Transaction validation is founded on public access to the transaction history. While the physical identity of the owner of address A (Alice) cannot be directly deduced from this, any observer can see that the same individual performed a given set of transactions. Even worse, although Alice can always create a new address A' she might have to transfer money from her old one to the new one in order to use it, at which point an external observer can make an educated guess that links A' to the owner of A. Indeed, there has been a growing amount of literature[1,23,30,31] showing how the transaction graph can be used to link together addresses. This, combined with external information (for example, vendor purchases) and further heuristic analysis, can lead to user identification. Currently, there even exist Bitcoin tracking companies (for example, Elliptic[a] and Chainalysis[b]) that monitor the body of transactions, aiming at identifying illicit activity.

Network-level deanonymization. By default, the Bitcoin peer-to-peer protocol does not protect the IP addresses of the participants since they are communicated in the clear. Researchers[4] have shown how this information can be used to deduce user identities. However, most Bitcoin client implementations can be configured to run over an anonymous Tor proxy, hiding the participants' addresses. Unlike what one might expect, this approach does not solve the problem. Subsequent work[5] has demonstrated how the interaction between Bitcoin and Tor can be exploited by an adversary who not only compromises user privacy (negating the anonymizing effect of the latter) but can also launch a stealthy man-in-the-middle attack, targeting the security of the Bitcoin protocol itself. While Biyukov[5] discusses a number of partial countermeasures, to the best of our knowledge there is no definitive way to protect the network-level anonymity of Bitcoin users yet. One likely candidate solution is the evolution of Bitcoin to operate over a tailor-made anonymity network that will not suffer from the issues discussed here. It should be noted that none of the techniques we mention in the sequel addresses network anonymity explicitly.

Real-world anonymity. We stress the gap between anonymity as a property of the cryptocurrency protocol execution and "real-world anonymity." For example, when one uses a cryptocurrency to purchase goods or services from a vendor they must provide the latter with certain personal information (identity for registration, physical address for delivery, email for pur-

Figure 1. Example of centralized mixing with four participants and a trusted mixer. No observer can "link" input to output addresses.



a https://www.elliptic.co
b https://www.chainalysis.com

chase confirmation, and so on). Thus, the vendor can trivially link the public key with its owner, in a strong sense. Moreover, this information may be extracted by others (for example, in case the vendor is hacked or a government agency issues a subpoena). Combined with "Know-your-Customer" anti-money laundering policies that enforce the collection of such data (like the one included in the USA Patriot Act of 2001) this can seriously compromise the privacy of cryptocurrency users.

Bitcoin Mixing

As discussed, a Bitcoin address can be potentially mapped to a physical entity by examining its related history of transactions (namely edges on the transaction graph) that are stored on the publicly accessible blockchain. This has prompted researchers to introduce various techniques for achieving anonymity.[22] One such prominent approach is *Bitcoin mixing* (or *Bitcoin tumbling*).

Suppose each one of the addresses A, B, C, and D wish to send one bitcoin to addresses A', B', C', and D' respectively. If these transactions are posted directly on the blockchain, everybody can deduce exactly how money flows. Bitcoin mixing "mixes" these transactions so the amount of information that becomes public is minimized—with Bitcoin mixing one would just find out that A's bitcoin went to one of A', B', C', or D', but not to which address exactly. The simplest way to achieve that is to use a trusted mixer (as we will discuss) who first receives the money from A, B, C, and D and then sends the money to A', B', C', and D' respectively. Clearly such an approach does not reveal information about the exact transaction edges. In order for this process to truly hide the link between input and output addresses, all users must participate with the same amount. (One can always use a larger amount and specify a fresh "change" address.) This provides privacy similar to k-anonymity[36] (assuming k participants) since no observer can distinguish which coins end up at each recipient.

A Bitcoin address can be potentially mapped to a physical entity by examining its related history of transactions that are stored on the publicly accessible blockchain.

Bitcoin mixing methods. There are various ways of Bitcoin mixing, achieving different levels of privacy, security, and efficiency. One key distinction has to do with how the parties that participate are coordinated. In theory, it is always possible for a party that wants to mix its coins to find a friend with similar goals and coordinate the exchange of some amount of bitcoins via an out-of-bound channel (for example, phone). This is a valid solution but in order to truly improve their privacy, users should try to hide inside a set of parties that is as large as possible. On the other hand, point-to-point coordination of hundreds or thousands of users can be very impractical, especially if the execution of the mixing protocol requires multiple rounds of communication. Therefore, many centralized solutions have been proposed where a third-party server, that receives a mixing fee, is utilized to handle the logistics of the transaction, under varying threat models (fully trusted, accountable, or untrusted). Finally, one must consider whether or not the identities of the mixing participants (or even the link between sender and recipient) will be revealed to other participants.

Centralized mixers. The simplest and easiest way to implement a form of Bitcoin mixing is via a trusted third party that serves as the *mixer* (shown in Figure 1). To send an amount of bitcoins from an address A to another address A', A first performs a transaction transferring a fixed amount to the mixer and sends an encryption of A' under the mixer's public key to the latter. After collecting a number of such transactions (assuming the same amount in each transaction) from multiple users—or, alternatively, after a certain amount of time has elapsed—the mixer sends, in a single Bitcoin transaction containing the recipients' addresses in a randomly permuted order, the same amount back to recipients' addresses. This achieves k-anonymity for a set that is as large as the number of parties that use the mixer within the given time increment, as there is no way for an external observer to distinguish the mapping between input and output addresses. The anonymity set can be further increased beyond the number of parties that use the mixer in the given time increment by sequentially mixing the coins multiple times (using several mix transactions), at the cost of reduced efficiency. One thing to note is this approach does not hide the fact these users used the mixer (and may, therefore, have "something to hide").

There exist multiple providers (for example, Bitmixer,[c] Bitlaunder,[d] Helix[e]) that offer this service for a small mixing fee with varying degrees of adoption. However, the most notable problem is that this approach requires "blindly trusting" the mixer. What if the mixer goes out of business? What if it is forced (for example, via a subpoena) to reveal the actual transaction links? Most importantly, what if it simply steals the coins? All these are valid issues and have indeed been, to some extent, observed in practice (for example, see Möser[27]).

Avoiding coin theft by the mixer. To mitigate the problem of coin theft by the mixer, Bonneau et al. proposed Mixcoin,[8] a Bitcoin mixer that holds the provider *accountable*. Theft is still possible but it can be reported via the use of signed *warrants*. In particular, before receiving A's coins, the mixer signs a statement of "*if A sends me x BTC by time t_1, I will send x'BTC back to B by time t_2*" (where x' is slightly smaller than x to account for a mixing fee) and sends this statement (with off-chain communication) to A. In case the mixer does not follow up on its end, A can publish this warrant damaging its reputation.

The first solution to truly avoid the possibility of coin theft was CoinSwap,[21] whose main building block is a timed-escrow protocol between two parties (also known as a 2-of-2 escrow). At a high level, a timed-escrow protocol that transfers money from Alice to Bob is implemented with the following transactions. The *initial* transaction is posted by Alice and places a number of bitcoins in escrow for a time window t. For Bob to claim these coins, a *release* transaction must be posted, signed by both Alice and Bob, before time t. Otherwise, the funds return to Alice. CoinSwap avoids coin theft by the mixer using two *correlated* timed-escrow protocols, one between the payer and the mixer and one between the mixer and the recipient, such that the recipient receives the money if and only if the mixer receives money from the sender. The downside of Coinswap is it requires multiple rounds of interaction and waiting for the validation of at least two blocks in the blockchain. Moreover, like Mixcoin, it also exposes the participants' identities to the mixer.

Hiding users' identities from the mixer. Blindcoin[37] is an extension to Mixcoin that utilizes *blind signatures* for the warrants. Blind signatures[11] operate like regular cryptographic signatures but allow a party to sign a message without knowing the message's exact content. A user A that wishes to mix her coins initially provides only a commitment to a fresh address she owns, "blinded" by a random value. After receiving the warrant from the mixer, A can remove the randomness and publish the address to a public log that contains all addresses to which the mixer must forward coins for that epoch. This completely hides the link between input and output address of a user even from the mixer itself, achieving full unlinkability. However, since Blindcoin builds on Mixcoin it can only offer a limited notion of security: a cheating mixer can steal a participant's coins, but the participant can prove that a theft took place thus damaging the mixer's reputation.

Achieving full security and unlinkability. A more recent proposal is TumbleBit,[16] which simultaneously achieves full unlinkability and avoids coin theft. TumbleBit merges techniques from secure two-party computation and zero-knowledge proofs in order to protect the user's privacy and the validity of the transaction (including enforcing the mixer to carry it out honestly). At the core of TumbleBit is the notion of an RSA *puzzle*. In order for a party A to anonymously send a number of bitcoins to party B (assuming B has just established a fresh public address) via a mixer M, the interaction proceeds in three phases as follows.

First, during an *escrow phase*, A posts an initial escrow transaction for a number of bitcoins to M and B contacts M requesting that M post a similar initial escrow transaction toward him. Assume that the signature that B needs from M in order to claim the escrowed value is σ. Moreover, B obtains from M (via an off-chain protocol execution) an RSA puzzle that consists of two values z, c. The former is $z = \varepsilon^e \bmod N$ for some ε where N, e is the public RSA key of M (that is, z is a deterministic RSA "encryption" of ε). Recall that without access to the corresponding RSA decryption key d, it is not possible for anyone to retrieve ε at this point. The key property is that the second part of the puzzle, c, is a symmetric key encryption of signature σ using ε as the key. That is, if B could decrypt the RSA encryption and retrieve ε, he would be able to use it to decrypt c, retrieve ε, and post the necessary release transaction to claim the bitcoins escrowed by M.

Then, during a payment phase, B will utilize A to get a solution for the puzzle. For this, B sends A a blinded version of z, by choosing randomness r and sending $z' = r^e z \bmod N$. Then A sends z' to M, asking him to provide a solution ε' for this version of the puzzle. M can do this easily, since he holds the decryption key d. (Note that M cannot link this interaction with B as z' is randomized and cannot be related to z.) This involves an interactive fair-exchange protocol between A and M which allows A to get the puzzle's solution while allowing M to obtain a release transaction for the escrow they set up during the previous phase, signed only by her. Finally, A sends ε to B who computes $\varepsilon = \varepsilon'/r$ and checks whether $\varepsilon^e = z \bmod N$ (in which case he "accepts" A's payment). The fair exchange protocol guarantees that A gets the solution to the RSA puzzle if-and-only-if M gets a release on the escrowed transaction.

Lastly, during a cash-out phase, M signs his part of the release transaction for the escrow A set up and B uses ε to retrieve the encrypted signature by M on their escrow, which he additionally signs himself. Both parties post the signed release escrow transactions claiming the escrowed values and this concludes the protocol, since the bitcoins "traveled" from A to B via M.

Assuming k sender/recipient pairs during a single TumbleBit epoch, all of which mix the same value, the anonymity property achieved by TumbleBit guarantees that M cannot deduce the corresponding sender for a given recipient, based on his entire epoch view (expect with probability $1/k$). To avoid leaking additional information based on the timing of different protocol phases, all mixing transaction phases are synchronized and take a predetermined amount of time. Moreover, the fair-exchange protocol guarantees that as

c https://bitmixer.io
d https://bitlaunder.com
e https://helix-light.com

soon as M provides a solution to the puzzle, he receives the information he needs to claim A's escrow. Finally, the properties of the fair exchange protocol also guarantee that this will only happen if M provides the correct solution which implies B is able to claim the escrow set up by M (note that A is always motivated to send the solution to B as the bitcoins she escrowed will be claimed by M even if she does not).

Peer-to-peer mixing solutions. Next we turn our attention to alternative approaches that obviate the need for an intermediate party. One obvious benefit of this approach is that it eliminates the need for mixing fees. Moreover, it is closer in spirit to the decentralized principle behind Bitcoin; if the participants can themselves perform this service, why rely on a central provider?

Mixing with a single transaction. Each Bitcoin transaction can contain multiple input and output addresses. This allows a user to join inputs from multiple addresses she owns in order to match the cost of a particular goal. For example, if Alice is required to transfer 5BTC to Bob as part of a purchase, Alice can combine 2BTC from one address she owns and 3BTC from another, as inputs to a transaction that transfers 5BTC to an address owned by Bob. However, the Bitcoin protocol does not explicitly require that all input addresses belong to the same party. Multiple parties can, in principle, contribute input addresses to the same transaction (as shown in Figure 2). CoinJoin[20] is a mixing approach proposed by Maxwell that takes advantage of this liberty that Bitcoin offers. A set of k users can agree to jointly create a transaction with k input addresses that transfers its inputs to k output addresses. Each party individually observes the transaction; if her own output address appears in the list of recipients, she signs the transaction as a payer with her private key. Eventually, the transaction carries k different signatures. This simple idea has served as the core of multiple subsequent implementations and optimizations.

Internal Unlinkability. While CoinJoin hides the shuffling of the coins from an outsider (thus providing *external unlikability*), participants trivially learn the mapping from input to output addresses (that is, it lacks *internal unlikability*). CoinShuffle[33] avoids this by utilizing an anonymous group communication protocol that can hide the participants' identities from each other. This is achieved with the simple trick of layered encryption, as shown in Figure 3 (for four parties).

Assume three parties A, B, and C, with corresponding public keys pk_A, pk_B, pk_C, that want to mix the same amount of bitcoins each by transferring them to addresses A', B', and C', respectively. A then encrypts A', in a layered manner, first under pk_C and then under pk_B, that is, computes $Enc_{pk_B}(Enc_{pk_C}(A'))$. Likewise, B encrypts B' under pk_C to get $Enc_{pk_C}(B')$. Then, A sends the encryption of A' to B who proceeds to remove the outer encryption layer (using her own decryption key), randomly shuffles the resulting encryption with her own encryption of B', and forwards both to C. At this point C receives A', B' encrypted under pk_C and has no way of guessing which belongs to whom. She simply decrypts these values, appends C', shuffles all of them and writes the transaction which is broadcast to all participants. Each one checks that her recipient address is in the receivers list and, if so, signs the transaction. Once all signatures are gathered, the transaction is published

Figure 2. Example of decentralized mixing with four participants. Only the parties learn the mapping from input to output addresses.

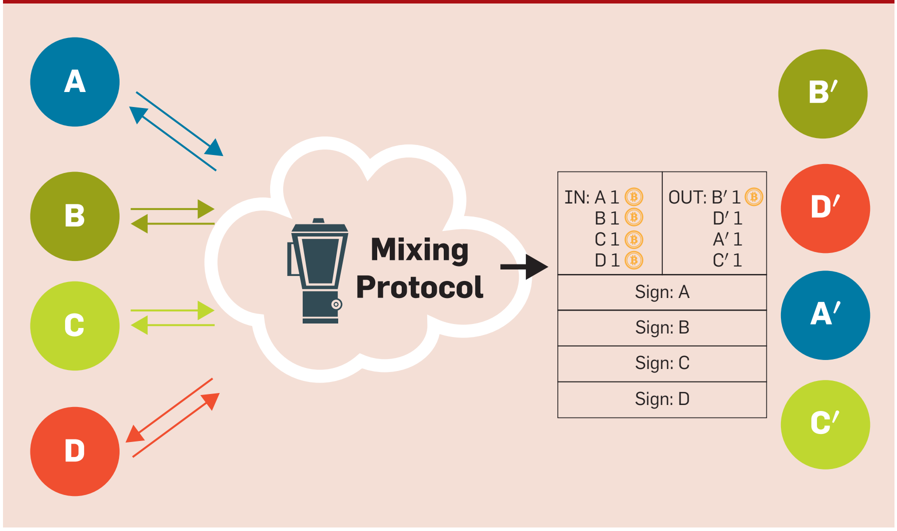
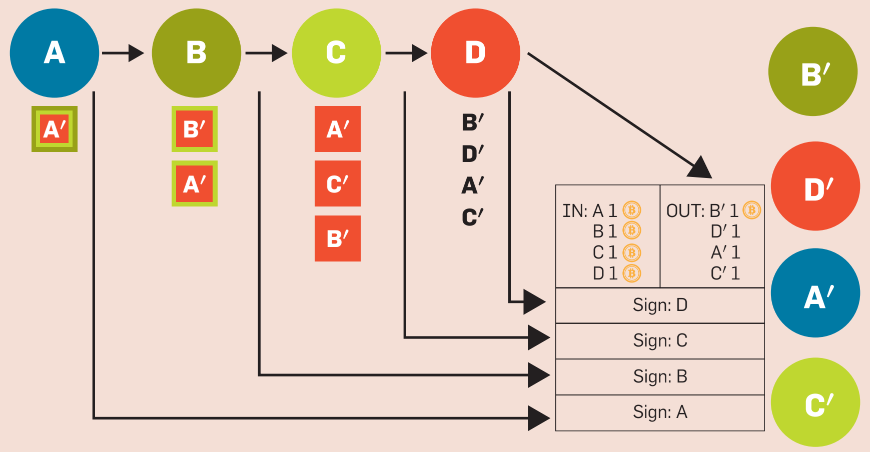


Figure 3. Decentralized mixing with CoinShuffle.

Each party encrypts her recipient address, in a layered manner, under the keys of all parties to her right and sends the shuffled vector of all addresses she sees to the next party who then removes the outermost layer (a colored rectangle denotes encryption with the key of the party with the corresponding color). The final party compiles the Bitcoin transaction and posts it on the blockchain. Everyone checks their recipient address is included and signs the transaction.



in the blockchain. CoinShuffle++[34] is an extension that uses a P2P network for traffic mixing while significantly reducing the performance and communication bandwidth.

Decentralized mixing with large anonymity sets. One issue with the peer-to-peer approaches is that their anonymity set is upper bounded by the number of participants in the mixing protocol, which is likely to be much smaller than that achieved by a "popular" centralized mixer (as we will discuss). One of the reasons is that typically the produced mixing transaction will have to carry a signature by each of the participants (for example, see figures 2 and 3). The total length of all these signatures blows up the size of the posted transaction significantly for larger sets, to the point that it may grow past the limits specified by Bitcoin (100KB for standard transactions). For example, Ruffing[34] is limited to 538 participants due to this.

In order to avoid this limitation, CoinParty[39] uses secure multiparty computation protocols that allow a set parties to collectively compute over their inputs in a way that does not reveal each party's input to other participating parties. Using such a protocol, the mixing participants collectively set up a single shared address (with off-chain communication) that is then used to transfer coins to fresh addresses. This means that the resulting transaction will only carry a single signature under this shared address. One major disadvantage of CoinParty is it requires at least 2/3 of the participants to be honest (which is an artifact of the secure multiparty protocol it uses), in order to guarantee no misbehavior with respect to the output signature.

Xim[6] can achieve large anonymity sets by an entirely different approach. Xim is a two-party mixing protocol that works as follows. First, during a pairing phase a party Alice that is interested in mixing her coins "advertises" this on the blockchain by posting a transaction that states she can be reached in a specific anonymous location (for example, a bulletin board maintained at a `.onion` Tor address she controls). An interested mixing partner Bob accesses the location expressing his interest by sending an anonymous location of his own (note that this communication takes place off the chain). After a specified amount of time, Alice chooses one of the interested partners that reached out to her (for example, Bob) and commits to proceeding by posting on her location a signed attestation of this. Within a fixed amount of time the two parties should post two transactions that "announce" their mutual pairing interest in a way that does not link their identities to outside observers. If that occurs, the two parties proceed to perform a single-transaction mixing, using a fair exchange protocol. If Bob backs down and does not post his transaction, Alice can simply announce she is looking for a new partner (without losing any funds) and if Alice backs down, Bob can post her signed attestation that confirms she changed her mind, "damaging" her reputation. Due to its interaction structure, Xim can achieve large anonymity sets, similar to the ones achieved by centralized mixers, assuming many participants are choosing to use it. The main downside is that it requires a significant blow-up in the end-to-end mixing time. A large portion of the communication happens sequentially over the chain itself therefore the waiting time for transactions to be collected by miners, added to blocks, posted to the chain, and substantially validated (by extending the chain) will typically be in the order of hours.

Alternative Privacy-Preserving Cryptocurrencies

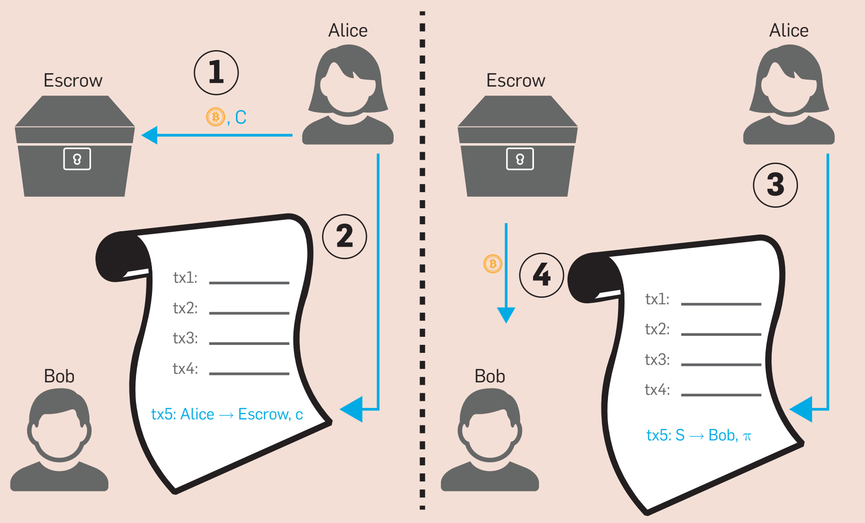
Here, we review some suggestions for alternative cryptocurrencies designed with the goal of providing stronger privacy guarantees than Bitcoin.

Privacy via ring signatures: CryptoNote. One of the first attempts to make transactions more private without additional interaction from the client (for example, using a mixing service or protocol) is CryptoNote,[38] the core idea of which has subsequently been refined and adopted in other currencies, for example, Bytecoin,[f] and Monero.[29] Like Bitcoin, every CryptoNote user has a public and a private key. Unlike Bitcoin however, the destination address of a transaction is a one-time public key, which is derived from the recipient's public key and some randomness chosen by the sender.

In particular, when Alice wants to send an amount m to Bob, she first establishes a one-time public key $pk_{B,r}$ with Bob using fresh randomness r and Bob's public key B. Then she posts a transaction on the blockchain that contains m, $pk_{B,r}$ and some pub-

Figure 4. Overview of Zerocoin.

(1–2) Alice places a coin with (hidden) serial number *S* and (visible) commitment *c* to escrow, by posting a corresponding transaction to the blockchain. (3–4) To pay Bob, Alice publishes a transaction with Bob as the receiver but no explicit sender. Instead of the sender, the transaction reveals *S* and a proof that it matches some coin in escrow. Everyone can check the validity of π but nobody can link the transaction to Alice.



f https://bytecoin.org

lic transaction information p. Next, Alice must sign the transaction in a way that proves her ownership of the funds being transferred but does not reveal her identity. Indeed, in order to do this Alice signs the transaction using the one-time secret key $sk_{A,r'}$ established during the transaction through which she originally acquired the funds. While this proves Alice's ownership over the transferred funds (since Alice is the only one that knows $sk_{A,r'}$, using regular digital signatures for the signature would trivially link the two transactions.

To prevent such leakage, CryptoNote uses ring signatures[12] to verify that a specific message was signed by some user belonging to a group of users, without revealing the signer's specific identity. Thus, Alice creates a transaction (which includes $pk_{B,r}$ as well as additional information p that will allow Bob to recover $sk_{B,r'}$ and choses some set of public keys PK which includes $pk_{A,r}$ from the public ledger. Alice then signs the transaction using $sk_{A,r'}$ and publishes the transaction and PK on a public ledger, thereby proving her ownership over the coin. Due to the hiding property of ring signatures, the signature may have originated from any of the users in PK. Thus, Alice is able to control the anonymity level of her transaction with Bob by simply varying the size of the set PK.

Zero-knowledge transactions: Zerocoin. As described earlier, the anonymity level provided by CryptoNote is directly related to the size of the set PK. However, the size of PK also affects the amount of work required by Alice in order to perform a transaction, thus hampering the performance of the cryptocurrency. Alleviating this issue, Zerocoin[25] uses a different approach that decouples the amount of work required by Alice from the achieved level of anonymity. Zerocoin works as an "overlay" over the Bitcoin protocol as follows: Assume Alice wishes to spend a (predetermined) number of bitcoins privately, without revealing her identity. The first step she takes is to *mint* a zerocoin by generating a random serial number S and by creating a commitment c to S using randomness r. Alice then publishes a transaction (for the amount she wishes to spend) from her address using c as destination (at this point, c can be seen as being held in escrow). The commitment c is then added by the network to a global, publicly visible set C of minted coins.

When Alice wishes to spend her new zerocoin, she creates a noninteractive zero-knowledge-proof-of-knowledge (NIZKPoK)[35,g] proof π of the statement "S is a valid opening to some commitment on an unspent zerocoin currently being held in escrow." Next, Alice publishes a transaction with Bob's address as destination and with an empty origin address containing π and S. At this point, due to the zero-knowledge property of π, there is no way to link Alice to any specific zerocoin commitment c.[h] The network accepts this transaction published by Alice only if the validation of π succeeds and S has not been previously spent. In this case, participants add S to the list of previously spent coins (see Figure 4).

Practical considerations. To minimize the size of the proof π, Zerocoin[25] implements the coin set C as an *accumulator*,[9] which is a cryptographic construction that allows efficient insertions and proofs of membership. Still, each spending transaction is 48KB (for 128-bit security level), exceeding the 10KB current limit for Bitcoin transactions. Also, note that the Bitcoin's source code does not support the necessary cryptographic operations.

Transactions with zk-SNARKs: Zerocash. Zerocash[2] is an alternative cryptocurrency that, unlike Zerocoin, hides both origin and destination addresses. Compared to Zerocoin, it provides additional functionality, that is, it handles transactions of arbitrary denominations, and it provides a way to give "change" after a transaction. Moreover, it improves Zerocoin's verification efficiency and proof size.

Protocol overview. Similar to Bitcoin, a Zerocash user Alice has a Zerocash address consisting of a public and secret key pair (pk_A, sk_A). Similar to Zerocoin, a coin c of value v is minted by having Alice sample a random serial number S and compute a commitment to the coin's value, serial number, and her public key pk_A. Next, Alice publishes a mint transaction that sends v bitcoins to the previously computed commitment c. As a result, the coin is being held in escrow and can only be spent by a user that knows Alice's secret key sk_A.

When Alice now wants to send v coins to Bob, she performs a *pour* transaction that is somewhat similar to the mint transaction: she posts a new transaction with a new coin c' with serial number S' but this time she ties c' it to Bob's public key pk_B; and she does not reveal her public address. Next, she computes a zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK)[13] proof π to the following claim: "(1) S is a valid opening to some unspent coin c tied to an address pk_A currently held in escrow; (2) I know the secret key sk_A corresponding to pk_A; (3) c' has the same value v as c." Alice publishes a zerocash transaction containing S, π, c' without mentioning Bob's public address. The network accepts Alice's transaction only if π verifies and S has not been previously spent. In this case, participants add S to the list of previously spent coins. Notice that unlike Zerocoin, Bob's public address is not included as part of Alice's transaction. In fact, the only information that ever appears in the ledger in plaintext is the serial number of spent coins. Monitoring the ledger, Bob can test if a new coin c' was sent to him by testing it using his secret key sk_B. At that point, Bob can spend the coin as he wishes.

Implementing the set of committed coins. Zerocash does not use an accumulator[9] for the set of committed coins. Instead it uses Merkle hash trees[24] along with zk-SNARKs proofs.[13] Merkle trees have the same interface with RSA accumulators (they allow efficient insertion of elements and proofs of membership) but can be encoded in a zk-SNARK proof much more efficiently when a "SNARK-friendly" collision-resistant hash function is used.

Practical considerations. The use of zk-SNARKs drastically changes the performance of Zerocash from that of Zerocoin. In particular, the spending transaction size is reduced to under 1KB and its verification time is less than 6ms. On the other hand, creating this transaction takes significantly longer as the zk-SNARK prover algorithm is particularly demanding (however, this may be smaller than the block creation

g NIZKPoKs are cryptographic systems similar to zk-SNARKs but achieving weaker performance guarantees for the verifier.

h Recall that unlike a regular Bitcoin transaction, Alice did not publish her identity and the transaction's sender.

time). In October 2016, Zcash[i]—a cryptocurrency based on Zerocash—was officially launched. As of Jan. 26, 2017, Zcash has a market capitalization of $20.5 million. It uses a mining mechanism similar to that of Bitcoin but based on an alternative memory-hard proof-of-work function[3] and it has four times smaller expected block creation time. The developers of Zcash chose to establish the public parameters upon which its security is bootstrapped via a secure multiparty computation protocol executed with a ceremony held among remote practitioners (some of which remained anonymous) and with several defense mechanisms deployed.[j]

Privacy beyond transactions: Hawk. The cryptocurrencies discussed so far aim to provide a single, basic functionality: transferring funds from Alice to Bob. However, imagine we had to implement a more complicated *contract* to decide how money would flow. For example, consider a Vickrey auction for some item offered by a seller S, where the transfer of money from Alice or Bob to S would depend on who made the highest bid. That person would finally take the item and pay the second highest price to S. Ethereum[k] is an alternative cryptocurrency aimed at securely executing such smart contracts on top of a blockchain-like public ledger. Unfortunately, Ethereum offers very weak privacy guarantees, revealing the sender's and receiver's addresses as well as all information and internal values computed inside the smart contract (in the example here, the bids of each user would be eventually leaked). Hawk[19] aims to offer notions of privacy while preserving arbitrary smart-contract functionality. The main protocol involves a party called the *manager* who is trusted for keeping participants' values (bids) secret, but not for executing the contract correctly.

At a high level, the protocol starts by having Alice and Bob *mint* a certain number of Hawk coins, say h_a and h_b, as in Zerocash and Zerocoin. Then, to participate in the auction there is a bidding period where Alice and Bob commit to their bids x_a and x_b using a hiding commitment, also computing a zk-SNARK proof they have minted enough coins to support their bids. When the bidding period ends, Alice and Bob post an encryption of their plaintext bids on the blockchain under the manager's public key, along with a zk-SNARK proof they have encrypted the same value as they committed in the bidding phase. Then the manager retrieves the plaintext values x_a and x_b, and redistributes the money based on the user-defined program that it executed privately, for example, an auction in this case, so the money goes from the winner to the seller, without leaking any information to the public. The manager submits a zk-SNARK proof indicating the correct execution of the private auction program, and the correct redistribution of money based on the private output of the program. Finally, the seller gets the new coins but nobody with access to the blockchain can find out who the winner was (assuming the manager does not leak the bids when running the auction).

In terms of concrete performance, assuming an auction with 100 participants, each one needs to publish two separate statements in the blockchain in preparation for the auction. The manager then publishes a final statement that concludes the auction. Each participant spends approximately 35sec preparing these statements in a phase that requires 4GB in memory. The corresponding costs for the manager are 3 minutes and 27GB.

Comparison of Existing Schemes

Next we attempt a comparison of the approaches discussed so far.

Mixing services. First, we compare mixing schemes in terms of their features (see the accompanying table, which is largely based on a similar comparison from Heilman[16]). We note that all of them are fully compatible with Bitcoin and do not require any modification in the codebase.

Decentralized protocols on the one hand avoid the need for a third party that in practice may become a single point of failure. However, they have the added issue of requiring participant coordination ahead of time in order to identify peers and form transactions. Also, the communication cost often scales quadratically in the number of participants, which in practice significantly limits the size of the anonymity set. For instance, none of CoinShuffle, CoinParty, or CoinShuffle++ scale the experimental evaluation they provide to more than 50 participants. Moreover, decentralized approaches are likely to achieve a quantitatively weaker privacy notion than the centralized solutions as, in contrast to the latter that hide an output address within the set of all mixer clients (input addresses) for a given time period, the

i https://z.cash
j https://goo.gl/fmHqUk
k https://www.ethereum.org

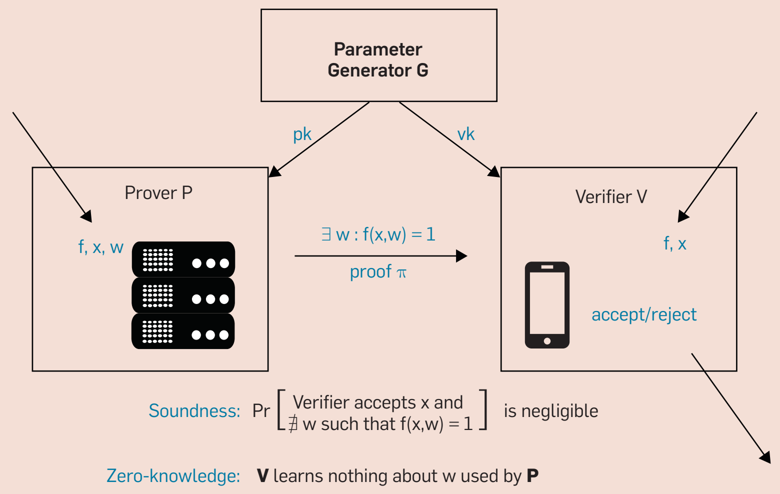
Comparison of the features of existing mixing schemes.

One bitcoin denotes a scheme fully achieves a property.
A parenthesis after an × in Unlinkable denotes which parties learn the link between input and output addresses.

	Avoids Coin-theft	Unlinkable	Anonymity Set	Adopted in practice
Untrusted mixer	×	× (mixer)	large	Bitmixer, Bitlaunder, Helix
Mixcoin[8]	accountable	× (mixer)	large	×
BlindCoin[37]	accountable	₿	large	×
CoinSwap[21]	₿	× (mixer)	large	×
TumbleBit[16]	₿ ₿	₿	large	Stratis[a]
CoinJoin[20]	₿	× (internal)	small	JoinMarket, DarkWallet, SharedCoin, Dash
CoinParty[39]	2/3 honest	₿	large	×
CoinShuffle[33]	₿	₿	small	Shufflepuff[b]
Coinshuffle++[34]	₿	₿	small	×
Xim[6]	₿	₿	large	×

a https://goo.gl/HXcr4J
b https://goo.gl/cCS2jz

The zk-SNARK Protocol


Parameter Generator G
pk
vk
Prover P
f, x, w
∃ w : f(x,w) = 1
proof π
Verifier V
f, x
accept/reject
Soundness: Pr [Verifier accepts x and ∄ w such that f(x,w) = 1] is negligible
Zero-knowledge: V learns nothing about w used by P

A *zero-knowledge succinct non-interactive argument of knowledge* (zk-SNARK)[13] is a protocol that allows a prover to prove claims of the form "I know *w* such that the output of program *P* on input *x*, *w* is 1" for pre-agreed program *P*. Crucially, the time it takes the verifier to check the validity of the prover's claim is much smaller than the time to run *P*(*x*,*w*). Moreover, during this verification process a polynomial-time verifier learns almost no information about *w* and the proof size generated by the zk-SNARK is short (for example, 288 bytes for Zerocash). While zk-SNARKs can be used to verify the execution of arbitrary programs, they have one notable downside. The public parameters used for proof construction and verification must be generated in a preprocessing phase by a trusted party. This raises the question of who can be entrusted to generate (and "forget") these parameters and opens a window of opportunity for an attacker to compromise the security of the system.

former hide it only within the set of participants of the particular transaction, which will typically be smaller (with the exception of Xim and Coinparty). The need to achieve larger anonymity sets (restricted only by the maximum transaction size and the hardness of coordinating) has given rise to services that "connect" interested users (for example, JoinMarket[l] for CoinJoin) operating as public bulletin boards and support for CoinJoin by existing wallets (for example, SharedCoin[m] and Darkwallet[n]). Note that the former was integrated to the popular blockchain.info wallet but support for it has since been suspended, partially due to issues related with limited privacy.[o] Finally, the idea behind CoinJoin has served as the core of for the alternative cryptocurrency Dash[p] that achieves large anonymity sets.

Sybil attacks (where an attacker poses as multiple mixing users in order to reduce the size of the anonymity set of honest participants) are a common problem for the above proposals. One partial countermeasure is imposing a "participation fee" that is payable by every user that wishes to mix her coins.[6,8,16,21,37,39] Finally, one technique that can be applied on top of some of these schemes (for example, Ruffing[32]) in order to hide the amount exchanged in the transaction is Confidential Transactions.[q]

Alternative cryptocurrencies. Among the cryptocurrencies we reviewed in this article, there exist two notable trade-offs. The privacy provided by CryptoNote to the transaction sender (Alice) directly depends on the size of the group Alice choses to participate in her ring signature. Moreover, Alice must publish the public keys of all the chosen group members. Thus, in order to remain *completely* undetectable Alice must use the public keys of all the users for her ring signature, making signature creation and verification expensive. On the other hand, Zerocoin and Zerocash achieve by default the "maximal" anonymity set, as all transactions ever to be published seem identical. However, they come with a drawback of their own. Zerocoin and Zerocash require a trusted party in order to setup the public parameters (for example, the RSA modulo for the former and the zk-SNARK parameters for the latter). While this only takes place once, as discussed above, any successful attack on the trusted party (including the party itself misbehaving) results in a complete compromise of the coins' security. Finally, Zerocash is much more efficient than Zerocoin, however it relies on much stronger "nonfalsifiable" cryptographic assumptions.[14]

Overall comparison. Attempting to compare these two "classes" of privacy techniques, one major drawback of mixing-based privacy solutions is they require various degrees of interaction from the client (either with the mixer or with other clients) in order to ensure privacy, which may impair their practical adoption. However, these solutions run on top of the widely used Bitcoin. On the other hand, any alternative cryptocurrency requires a significant amount of time for the community to become familiar with as well as to test it and trust it. As most of these protocols require a large crowd-base size in order to achieve strong security properties, this becomes an inhibiting factor for every new proposal. The main advantage of privacy-preserving cryptocurrencies is they increase the client's anonymity set from relatively small sets of clients that use a particular mixing service or participate in a transaction, to large sets that include *all* the users of a given cryptocurrency.

Discussion

We believe our exposition so far indicates there is no general consensus regarding a technique for anonymous cryptocurrencies. This should come as no surprise given the relative infancy of the field and the fact that different participants may have different privacy requirements. For example, for most users it may be sufficient to run a single round of CoinJoin with a dozen users whereas privacy-aware users

l https://github.com/JoinMarket-Org/joinmarket
m https://en.bitcoin.it/wiki/Shared_coin
n https://www.darkwallet.is
o http://www.coinjoinsudoku.com
p https://www.dash.org
q https://bitcointalk.org/index.php?topic=1085273

may choose to opt for something more thorough. Moreover, there exist other approaches for privacy that do not fall within any of the two categories, for example, private payments in credit networks[26] and payment channels.[15,r] Next, we discuss a number of open problems that arise while trying to design better private cryptocurrencies.

Unified formal privacy definition. One particular issue has to do with the formal treatment of the problem. While some existing works attempt to provide a definition of anonymity in the context of cryptocurrencies (for example, Bonneau[8] and Meiklejohn[22] for mixers and Ben-Sasson[2] and Miers[25] for alternative cryptocurrencies), there is no de facto unified privacy definition that would allow a fair comparison of different proposals (for example, it is difficult to quantitatively compare the security properties of Zerocash and Cryptonote if they satisfy different privacy definitions). Due to the nature and scale of cryptocurrency implementations, one very robust (but challenging in formulation) framework would be that of universal composability,[10] along the lines of the one introduced in Kosba[19] for private smart contracts.

Strong anonymity with milder sssumptions. A more concrete problem has to do with designing cryptocurrencies that achieve the strong anonymity levels of Zerocash but without the need for a sensitive trusted setup phase and without relying on the non-falsifiable cryptographic assumptions inherent to zk-SNARKs. The problem becomes even more important in the context of smart contracts as Hawk requires a separate trusted setup process for the generation of each different contract.

Scalable anonymous cryptocurrencies. Perhaps the most important challenge for Bitcoin (and other cryptocurrencies) is scalability; for any privacy solution to be widely used in practice, it must not only protect the users' anonymity but also be able to scale to realistic numbers of users and transactions. For example, Zerocash[2] reports more than 40 seconds of proving time per transaction and requires approximately 1GB of memory. Both of these inhibit the potential of large-scale deployments.

Privacy abuse and stricter policies. While the goal of this article has been to provide an overview of techniques for achieving anonymity in cryptocurrencies, it should be noted that increased user privacy may raise concerns, such as users participating in illegal activities[18] or facilitating various cryptographic ransomware.[s] This in turn may lead to stricter government regulation of cryptocurrency transactions[t] and requests for auditability,[u] which seems inherently incompatible with the need for stronger user anonymity.

Acknowledgments

This work was supported in part by NSF awards #1514261 and #1652259, financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology, the 2017-2018 Rothschild Postdoctoral Fellowship, and the Defense Advanced Research Project Agency (DARPA) under Contract #FA8650-16-C-7622. C

r For detailed presentation, see https://z.cash/static/R3_Confidentiality_and_Privacy_Report.pdf

s https://goo.gl/8Eujkr
t https://goo.gl/4WueYR
u https://goo.gl/iUnNMQ

References

1. Androulaki, E., Karame, G., Roeschlin, M., Scherer, T. and Capkun, S. Evaluating user privacy in Bitcoin. In *Proceedings of FC 2013*, 34–51.
2. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E. and Virza, M. Zerocash: Decentralized anonymous payments from Bitcoin. In *Proceedings of IEEE SP 2014*, 459–474.
3. Biryukov, A. and Khovratovich, D. Equihash: Asymmetric proof-of-work based on the Generalized Birthday Problem. In *Proceedings of NDSS 2016*.
4. Biryukov, A. and Khovratovich, D. and Pustogarov, I. Deanonymisation of clients in Bitcoin P2P network. In *Proceedings of ACM CCS 2014*, 15–29.
5. Biryukov, A. and Pustogarov, I. Bitcoin over Tor isn't a Good Idea. In *Proceedings of IEEE SP*, 2015, 122–134.
6. Bissias, G.D., Ozisik, A.P., Levine, B.N. and Liberatore, M. Sybil-resistant mixing for Bitcoin. In *Proceedings of WPES*, 2014, 149–158.
7. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A. and Felten, E.W. SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies. In *Proceedings of IEEE SP*, 2015, 104–121.
8. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A. and Felten, E.W. Mixcoin: Anonymity for Bitcoin with accountable mixes. In *Proceedings of FC*, 2014, 486–504.
9. Camenisch, J. and Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of CRYPTO*, 2002, 61–76.
10. Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of FOCS*, 2001, 136–145.
11. Chaum, D. Blind signatures for untraceable payments. In *Proceedings of CRYPTO '82*, 199–203.
12. Fujisaki, E. and Suzuki, K. Traceable ring signature. In *Proceedings of PKC*, 2007, 181–200.
13. Gennaro, R., Gentry, C., Parno, B. and Raykova, M. Quadratic span programs and succinct NIZKs without PCPs. In *Proceedings of EUROCRYPT*, 2013, 626–645.
14. Gentry, C. and Wichs, D. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of STOC*, 2011, 99–108.
15. Green, M.D. and Miers, I. Bolt: Anonymous payment channels for decentralized currencies. *IACR Cryptology ePrint Archive*, 2016, 701.
16. Heilman, E., Baldimtsi, F., Alshenibr, L., Scafuro, A. and Goldberg, S. TumbleBit: An untrusted tumbler for Bitcoin-compatible anonymous payments. In *Proceedings of NDSS*, 2017.
17. Hileman, G. and Rauchs, M. Global cryptocurrency benchmarking study. *Cambridge Centre for Alternative Finance Global Cryptocurrency Benchmarking Study*, 2017.
18. Juels A., Kosba, A. E., and Shi, E. The ring of Gyges: Investigating the future of criminal smart contracts. In *Proceedings of ACM CCS*, 2016, 283–295.
19. Kosba, A.E., Miller, A., Shi, E, Wen, Z., and Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Proceedings of IEEE SP*, 2016, 839–858.
20. Maxwell, G. CoinJoin: Bitcoin privacy for the real world. bitcointalk.org, Aug. 2013.
21. Maxwell, G. CoinSwap: Transaction graph disjoint trustless trading. bitcointalk.org, Oct. 2013.
22. Meiklejohn, S. and Orlandi, C. Privacy-enhancing overlays in Bitcoin. In *Proceedings of FC Workshops, BITCOIN, WAHC, and Wearable*, 2015, 127–141.
23. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S. A fistful of Bitcoins: Characterizing payments among men with no names. In *Proceedings of IMC*, 2013, 127–140.
24. Merkle, R.C. A certified digital signature. In *Proceedings of CRYPTO '89*, 218–238.
25. Miers, I., Garman, C., Green, M., and Rubin, A.D. Zerocoin: Anonymous distributed e-cash from Bitcoin. In *Proceedings of IEEE SP*, 2013, 397–411.
26. Moreno-Sanchez, P., Kate, A., Maffei, M., and Pecina, K. Privacy preserving payments in credit networks: Enabling trust with privacy in online marketplaces. In *Proceedings of NDSS*, 2015.
27. Möser, M. An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem. In *IEEE 2013 eCrime Researchers Summit*.
28. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008; http://bitcoin.org/bitcoin.pdf.
29. Noether, S., Mackenzie, A., and the Monero Research Lab. Ring confidential transactions. *Ledger 1* (2016) 1–18.
30. Reid, F. and Harrigan, M. An analysis of anonymity in the Bitcoin system. In *Proceedings of IEEE PASSAT and SocialCom*, 2011, 1318–1326.
31. Ron, D. and Shamir, A. Quantitative analysis of the full Bitcoin transaction graph. In *Proceedings of FC 2013*, 6–24.
32. Ruffing, T. and Moreno-Sanchez, P. Mixing confidential transactions: Comprehensive transaction privacy for bitcoin. *IACR Cryptology ePrint Archive*, 2017, 238.
33. Ruffing, T., Moreno-Sanchez, P., and Kate, A. CoinShuffle: Practical decentralized coin mixing for Bitcoin. In *Proceedings of ESORICS*, 2014.
34. Ruffing, T., Moreno-Sanchez, P., and Kate, A. P2P mixing and unlinkable Bitcoin transactions. In *Proceedings of NDSS*, 2017.
35. Sahai, A. Non-malleable non-interactive zero knowledge and adaptive chosen ciphertext security. In *Proceedings of FOCS '99*, 543–553.
36. Sweeney, L. k-Anonymity: A model for protecting privacy. *Intern. J. Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 5 (2002), 557–570.
37. Valenta, L. and Rowan, B. Blindcoin: Blinded, accountable mixes for Bitcoin. In *Proceedings of the 2015 FC International Workshops, BITCOIN, WAHC, and Wearable*, 112–126.
38. van Saberhagen, N. CryptoNote v 2.0; https://cryptonote.org/whitepaper.pdf.
39. Ziegeldorf, J. H., Grossmann, F., Henze, M., Inden, N., and Wehrle, K. CoinParty: Secure multi-party mixing of Bitcoins. *CODASPY* (2015), 75–86.

Daniel Genkin (danielg3@cis.upenn.edu) is a postdoctoral researcher at the University of Pennsylvania, Philadelphia, and the University of Maryland, College Park, MD, USA.

Dimitrios Papadopoulos (dipapado@cse.ust.hk) is an assistant professor of computer science and engineering at Hong Kong University of Science and Technology.

Charalampos Papamanthou (cpap@umd.edu) is an assistant professor of electrical and computer engineering at the University of Maryland, College Park, MD, USA.