Escrow protocols for cryptocurrencies: How to buy physical goods using Bitcoin

Steven Goldfeder[1], Joseph Bonneau[2], Rosario Gennaro[3], and Arvind Narayanan[1]

[1] Princeton University
{stevenag,arvindn}@cs.princeton.edu
[2] Stanford University and EFF
jbonneau@cs.stanford.edu
[3] City College, City University of New York
rosario@cs.ccny.cuny.edu

Abstract. We consider the problem of buying physical goods with cryptocurrencies. There is an inherent circular dependency: should be the buyer trust the seller and pay before receiving the goods or should the seller trust the buyer and ship the goods before receiving payment? This dilemma is addressed in practice using a third party escrow service. However, we show that naive escrow protocols introduce both privacy and security issues. We formalize the escrow problem and present a suite of schemes with improved security and privacy properties. Our schemes are compatible with Bitcoin and similar blockchain-based cryptocurrencies.

1 Introduction

While Bitcoin and its many successor cryptocurrencies offer a secure way to transfer ownership of coins, difficulty arises when when users wish to exchange digital assets for physical goods. At a high level, parties wish to perform an *atomic exchange* with *guaranteed fairness* — i.e. either both the currency and goods will change ownership or neither will. The same difficulty arises in electronic commerce with traditional payment mechanisms. A buyer doesn't want to pay without assurance that the seller will ship the purchased goods, while a seller doesn't want to ship without assurance that payment will be received

Traditionally, this problem is solved in one of two ways. For large retailers with significant reputation (e.g. Walmart or Overstock) most customers are sufficiently confident that goods will be shipped that they are willing to pay in advance. For smaller sellers without a global reputation, buyers typically pay via a trusted third party, such as eBay or Amazon. If the buyer does not receive the item or the transaction is otherwise disputed, the third party will mediate the dispute and refund the buyer if deemed necessary. In the interim, the funds are in escrow with the intermediary. When users pay with credit cards, credit card companies often serve a similar role. A buyer can register a complaint with their issuer who will mediate and reverse charges if fraud is suspected.

This model has been adapted for online marketplaces employing cryptocurrencies for payment, including the original Silk Road [23] and many successors. In this model, the buyer transfers the payment to a trusted third party who only transfers it to the seller once it has ascertained that the product was delivered. However, this approach is not optimal for two reasons. First, it requires the third party to be actively involved in every transaction, even when there is no dispute. Second, it is vulnerable to misbehavior by the mediator, which can simply pocket the buyer's payment and never transfer it to the seller. This is considerably more difficult to trace or rectify due to the irreversible and pseudonymous nature of Bitcoin transactions. Furthermore, the history of Bitcoin exchanges [39] and online marketplaces [23] has been plagued by fraud and hacks, making it difficult for buyers or sellers to place high trust in any single service as a trusted third party. While better escrow protocols, which do not allow the mediator to trivially abscond with funds, are known in the literature [20] (and in practice [7]), they still introduce a number of problems.

Fortunately, Bitcoin's scripting language enables better protocols than the ones currently in use. We define a series of desirable properties for escrow protocols to have:

Security. Intuitively, an escrow protocol is secure if the mediator(s) cannot transfer the funds to anyone other than the buyer or the seller.

Passivity. A passive escrow protocol requires no action on the part of the mediator if no dispute arises, making the common case efficient.

Privacy. If implemented naively, escrow transactions can leave a distinct fingerprint visible on the blockchain which can potentially leak sensitive business information. For example, online merchants may not want their competitors to learn the rate at which they enter disputes with customers. We define a series of privacy properties regarding whether observers can determine that an escrow protocol was used, if a dispute occurred, or how that dispute was resolved.

Group escrow. To reduce the risk of trusting any single party to adjudicate disputes honestly, we introduce the notion of *group escrow*, relying on a group of mediators chosen jointly by the buyer and seller. Group escrow schemes should not require communication between the mediators, leaving the buyer and seller free to assemble a new group in an ad-hoc manner. Cheating should only be possible if a majority of the mediators colludes with one of the transacting parties.

Our contributions. To our knowledge, we are the first to formally study the escrow problem for physical goods and define the related properties. We introduce a series schemes with various properties, building up to our group escrow schemes which is secure, private, and passive. We note that our protocol is fully compatible with Bitcoin today as well as most other blockchain-based cryptocurrencies.

2 Background and Tools

2.1 ECDSA

Bitcoin, along with most subsequent cryptocurrencies, employs the Elliptic Curve Digital Signature Algorithm (ECDSA) using NIST's `secp256k1` curve [2–4]. Our protocols in this paper are largely agnostic to the details of the signature algorithm, but we include a summary of ECDSA in Appendix A.2 for completeness.

2.2 Secret sharing and threshold cryptography

Threshold secret sharing is a way to split a secret value into shares that can be given to different participants, or players, with two properties: (1) any subset of shares can reconstruct the secret, as long as the size of the subset equals or exceeds a specified threshold (2) any subset of shares smaller than this threshold together yields *no information* about the secret. In the most popular scheme, due to Shamir, the secret is encoded as a degree t polynomial, and a random point on this polynomial is given to each of n players, any $t + 1$ of which can be used to precisely reconstruct the polynomial using Lagrange interpolation [42].

Secret sharing schemes are fundamentally one-time use in that once the secret is reconstructed, it is known to those who participated in reconstructing it. A more general approach is *threshold cryptography*, whereby a sufficient quorum of participants can agree to use a secret to execute a cryptographic computation without necessarily reconstructing the secret in the process. A (t, n)-*threshold signature* scheme distributes signing power to n players such that any group of at least $t + 1$ players can generate a signature, whereas a smaller group cannot.

A key property of threshold signatures is that the private key need not ever be reconstructed. Even after repeated signing, nobody learns any information about the private key that would allow them to produce signatures without a threshold sized group. Indeed, threshold cryptography is a specific case which led to the more general development of secure multiparty computation [30].

2.3 ECDSA threshold signatures

Gennaro *et al.* presented an ECDSA threshold signature scheme in [27]. While previous threshold DSA signature schemes existed in the literature [28, 29, 35], the scheme in [27] is the only ECDSA scheme that works for arbitrary n and any $t < n$. All of our constructions that use threshold signatures can be instantiated with the scheme from [27]. The shared key in this scheme can either be distributed by a trusted dealer or generated jointly by the participating parties in a trustless manner.

2.4 Stealth addresses and blinded addresses

Bitcoin stealth addresses [6] are a special address type that solve the following problem: Alice would like to publish a static address to which people can send money. While she can do this, a blockchain observer will be able to trace all incoming payments to Alice. Alice would thus like to publish such an address while ensuring that the incoming payments she receives are neither linkable to her nor to each other. Moreover, the payer and Alice should not need to have any off-blockchain communication.

While stealth addresses are an elegant solution to this problem, these addresses have a unique structure, and as a result, the anonymity set provided by using stealth addresses is limited to the set of Bitcoin users that use them. For the use cases in this paper, we can relax the requirement that all communication must take place on the blockchain. Indeed, when there is a dispute in an escrow transaction, we expect that the parties will communicate offline with the mediator. Thus, we allow offline communication and as a result are able to extend the anonymity set provided by such addresses to all Pay-to-PubkeyHash transactions on the blockchain.

Our basic technique is largely the same as the one used in Bitcoin stealth addresses [6] as well as deterministic wallets [43]. However, to our knowledge, we are the first to prove its security. We present the details here, and refer to these addresses as *blinded addresses*.

An ECDSA key pair is a private key $x \in Z_q$ and a public key $y = g^x$ computed in $\mathcal{G}$ (see Appendix A.2). For a given ECDSA key pair, (y, x), we show a blinding algorithm that has the following property: anybody that knows just the public key y can create a new public key y' and an auxiliary secret $\hat{x}$ such that in order to create a signature over a message with the key y', one needs to know both the original key x as well as the auxiliary secret $\hat{x}$. We stress that the input to this algorithm is only the public parameters and the public key y.

On input y:

- choose $\hat{x} \in Z_q$ at random
- compute $\hat{y} = g^{\hat{x}}$ in $\mathcal{G}$
- compute $y' = y \cdot \hat{y}$ in $\mathcal{G}$
- output blinded public key y' and auxiliary secret $\hat{x}$

One who knows both x and $\hat{x}$ can create a signature that will verify with the public key y'. This is clear as the private key corresponding to y' is simply $x' = x + \hat{x}$ in $\mathcal{G}$.

See Appendix B for a security and privacy argument for this blinding scheme.

3 Related Work

3.1 Fair Exchange

The problem of fair exchange is how two mutually distrusting parties can jointly exchange digital commodities such that both parties receive the other party's input or neither do. Indeed, fair exchange is a special case of fair two-party computation in which two parties wish to jointly perform a function over private inputs such that either both parties receive the output or neither does.

Fair exchange has been studied extensively in the cryptographic literature [13, 17, 31]. Blum [19] and later Bahreman *et al.* [15] studied the problem of contract signing and how to send certified electronic mail – that is one party sends a document and the other sends a receipt– in a fair manner. Jakobsson studied fair exchange in the context of electronic checks [31].

The study of fair exchange naturally leads to the desire for *optimistic* (or *passive*) protocols [13] in which the third party only gets involved when there is a dispute. Such protocols are ideal in that they are far easier to use at scale as presumably the majority of transactions will not be disputed. This model is often used in designing fair protocols[14, 21, 26, 33, 38].

One approach to building fair two-party protocols uses monetary penalties [33, 34]. Intuitively, parties are incentivized to complete the protocol fairly, and if one party receives its output but aborts before the other party does, the cheating party will have to pay a penalty. Recently, several papers have proposed variations of this idea and shown how one could build secure protocols on top of Bitcoin, crafting Bitcoin transactions in such a way that a cheater will be automatically penalized or an honest party rewarded [11, 12, 18].

3.2 Exchanging bitcoins for digital goods

Exchanging units of a cryptocurrency for a digital good can be thought of as a special case of fair exchange. Elegant protocols exist which facilitate cross-currency exchange in a fair and trustless manner [1, 10, 20, 24].

Zero-knowledge contingent payments (ZKCP) solves the problem of using Bitcoin to purchase a solution to an NP-problem. Maxwell first presented the ZKCP protocol in 2011 [37] and it was publicly demonstrated in 2016 when bitcoins were traded for the solution to a Sudoku puzzle [36]. Banasiki *et al.* formalize and refine Maxwell's ZKCP protocol [16].

Juels *et al.* [32] propose a protocol for purchasing the private key to a specified public key, using a platform with a Turing-complete scripting language such as Ethereum. We limit our focus to the simpler capabilities of Bitcoin, which are sufficient to build protocols for escrowing payment for physical goods.

4 Escrow: Motivation, definitions, and model

Existing fair-exchange schemes apply only to the transfer of digital assets and generally fall into one of the following categories:

- Protocols that rely on transferring a digital signature (or "electronic check" [31]). These protocols give a trusted third party the ability to reconstruct the signature, thus assuring fairness.
- Protocols that rely on the fact that the digital asset can be reproduced and re-sent. Broadly, these schemes resolve disputes by enabling the mediator to reconstruct the desired asset. The disputing party in essence deposits a copy of its digital asset with the mediator.

Bitcoin breaks both of these assumptions, so none of the existing fair-exchange techniques work. While Bitcoin transactions are signed with digital signatures, they are fundamentally different from the electronic checks and other electronic forms of payment that are discussed in the existing literature. Under those schemes, the assumption was always that the transfer of the digitally signed transaction was equivalent to being paid. This was either because it relied on an older form of electronic cash in which the signed statement served as a bearer token and anyone bearing it could cash in the money, or it was due to the fact that the signed statement served as a contract, and one could take the contract to a court to receive payment.

With Bitcoin, however, a digitally signed transaction is insufficient since until the transaction is included in the blockchain, the buyer can double spend that transaction. Thus, bitcoins cannot be escrowed in the traditional manner. The buyer can sign a transaction that pays the seller, but this cannot be kept in escrow. If it is in escrow, the buyer can attempt to prevent its inclusion in the blockchain by *frontrunning.* That is, the buyer can quickly broadcast a conflicting transaction if he sees the escrowed transaction broadcast to the network. The only guaranteed way to know a signed transaction will have value is to include it in the blockchain—but at that point the seller has been paid.

Of course, if the buyer double spends, the seller can use the signed transaction to prove that fraud has occurred. But remember that Bitcoin does not use real-world identities and often parties interact anonymously, so proof of fraud will generally be insufficient to recover lost money.

From the seller's point of view, shipping physical goods is also unlike scenarios considered in the fair exchange literature because it is not possible to cryptographically prove that the seller behaved honestly. We assume our mediator will have to evaluate non-cryptographic evidence, such as package tracking numbers or sign-on-delivery receipts, as online merchants already do today.

The public nature of the Bitcoin blockchain – i.e. the entire transaction history of Bitcoin is public – is also distinct from traditional fair exchange assumptions. In a traditional fair exchange protocol, there is no global ledger so no outsider could learn anything about the exchange or the escrow transactions. With Bitcoin transactions, however, we will need to consider and actively protect against the privacy implications imposed by the public nature of the blockchain.

Thus, unlike existing protocols for fair-exchange, our goal in this paper is *not* to provide a cryptographic way to mediate disputes. Our goal is instead to develop techniques in which the transacting parties can *passively* and *privately* allow a third party to mediate their transaction. To achieve *fairness*, our protocols will make sure that both transacting parties cannot deviate from the semi-trusted third party's ruling.

4.1 Our scenario

Suppose Alice, a(n online) merchant, is selling an item to Bob, a (remote) customer. A natural dilemma arises: when should Bob pay? If Bob pays immediately, he runs the risk of Alice defrauding him and never sending him the item. Yet if he demands to receive the item before paying, Alice runs the risk of being defrauded by never being paid.

This problem arises in any payment system where the service and the payment cannot be simultaneously exchanged. A trivial solution is to use a payment platform which escrows the payment and can mediate in the event of a dispute. Reversible payment systems (e.g. credit card payments) enable the transaction to go through right away, but we still describe them as escrow services because the intermediary has the ability to undo the payment.

Because Bitcoin transactions are irreversible, we must rely on an explicit escrow service if the buyer and seller don't trust each other. Thus, instead of sending money to Alice directly, Bob sends the payment to a special *escrow address* that neither Bob nor Alice is able to withdraw from unilaterally. A *mediator* is a third-party used to mediate a transaction which is capable of deciding which party can withdraw funds from the escrow address.

4.2 Active and optimistic protocols

While a mediator must take action in the case of a dispute, we would like to avoid requiring any action by the mediator if no dispute arises. We define the requirements placed on the mediator with the following two properties:

Definition 4.1 (Active on deposit). *An escrow protocol is active on deposit if the mediator must actively participate when transacting parties deposit money into escrow.*

Definition 4.2 (Active on withdrawal). *An escrow protocol is active on withdrawal if the mediator must actively participate when transacting parties withdraw money from escrow even if there is no dispute.*

Of course a protocol may be both *active on deposit* and *active on withdrawal.* Note that the mediator is, by definition, always active in the event of a dispute, so we only consider the dispute-free case in our definition of active on withdrawal. Combining these two definitions, we can define the requirements for a mediator to be purely passive, or optimistic:

Definition 4.3 (Optimistic). *An escrow protocol is optimistic (eq. passive) if it is neither active on deposit nor active on withdrawal.*

4.3 Security of escrow protocols

While the essential nature of a mediator is that both parties must trust it to make a fair decision in the event of a dispute, we can consider the consequences if a mediator acts maliciously. We will consider only an *external* malicious mediator, meaning an adversary that does not also control one of the transacting parties. An *internal* malicious mediator also controls (or perhaps *is*) one of the participating parties. It is clear that security against an internal malicious mediator is unachievable. Recall that when a dispute arises, it is the responsibility of the mediator to award the funds to the correct party even if the losing party objects. Thus, any mediator by definition must have the ability to award the funds to one of the parties when both the mediator and that party cooperate. An internal adversary that controls the mediator as well as one of the transacting parties can create a dispute and have the mediator rule in its favor, guaranteeing that it receives the funds. For this reason, we define security of mediators only using the notion of an external attacker[4]:

Definition 4.4 (Secure). *An escrow protocol is secure if a malicious mediator cannot transfer any of the money held in escrow to an arbitrary address without the cooperation of either the buyer or seller.*

4.4 Privacy

Another concern for escrow protocols is privacy. The Bitcoin blockchain is public and reveals considerable information, including the amounts and addresses of all transactions. For escrow transactions, we consider three notions of privacy. An external observer is a party other than the transacting parties or the mediator.

Definition 4.5 (Dispute-hiding). *An escrow protocol is dispute-hiding if an external observer cannot tell whether there was a dispute that needed to be resolved by the mediator.*

Definition 4.6 (Externally-hiding). *An escrow protocol is externally-hiding if an external observer cannot determine which transactions on the blockchain are components of that escrow protocol.*

Note that our definition of externally hiding inherently relies on what baseline (non-escrow) transactions are occurring on the blockchain. For our purposes, we assume all non-escrow transactions are simple transactions sending money to a specified address (in Bitcoin parlance, a P2PKH transaction).

Definition 4.7 (Internally-hiding). *An escrow protocol is internally-hiding if the mediator itself cannot identify that the protocol has been executed with itself as a mediator in the absence of a dispute.*

We note that internally-hiding and externally-hiding are distinct properties and neither implies the other. Clearly, a scheme could be externally-hiding but not internally-hiding. This will occur if the mediator can tell that money has been put in its escrow, but an outsider looking at the blockchain cannot detect that escrow is being used. More interestingly though, a scheme can be internally-hiding but not externally-hiding. This occurs when it is clear from looking at the blockchain that escrow is being used, but the mediator cannot detect that its service is the one being used.

[4] There may be other desirable features that can be categorized as security properties that are out of the scope of this work.

It is clear why a company may want full privacy as they may want to keep all details of their business private. However, it is possible that an online merchant might not need its escrowed payments externally or internally hiding (say, it publicizes on its website that it uses escrow with a specific mediator). The company may however still want the escrow protocol to be dispute-hiding so that competitors cannot determine how often sales are disputed.

Of course, a buyer may take the exact opposite approach and demand transparency – i.e. that any company that it interacts with uses an escrow service that is not dispute-hiding so that the buyer can use this information to determine how often the seller's transactions are disputed.

4.5 Denial of Service

Our definition of security only prevents a directly profitable attack. Namely, the goal of the adversary is to steal some or all of the money being held in escrow by transferring the money elsewhere (e.g. to an address the adversary controls). However, a malicious mediator might instead deny service by refusing to mediate when there is a dispute.

The power of a denial-of-service attack is directly related to the type of mediator. For an active-on-withdrawal protocol, the denial-of-service attack can be launched even when the parties do not dispute, whereas for an optimistic protocol a denial-of-service attack can only be launched when the parties dispute.

Note that a denial-of-service attack may be profitable if the mediator is able to extort a bribe from the transacting parties in order to resolve a dispute. If the mediator suffers no loss if the dispute is never resolved, then it carries no financial risk from attempting such extortion. Of course, it may face significant risk to its reputation.

We can design schemes that prevent denial of service in Section 5.5, but as we will see, running such a service requires the mediator to put its own money into escrow as a surety bond and requires an active-on-deposit protocol.

5 Escrow protocols

In the previous section, we provided several definitions and security models outlining various types of mediators. We now propose several protocols for mediators and show which properties they fulfill; we refer the reader to Appendix C for detailed analysis of each scheme's properties.

5.1 Escrow via direct payment (the Silk Road scheme)

The simplest scheme is one in which the buyer sends money directly to the mediator's address. The mediator will then transfer the funds to the seller or back to the buyer as appropriate. In case of a dispute, the mediator will investigate and send the funds to the party that it deems to be correct. The illicit marketplace Silk Road famously used a variation of this method of escrow.

To improve privacy, rather than sending the funds to the mediator's long term address, the buyer can send funds to a blinded address (Section 2.4). This will allow the scheme to remain not-active-on-deposit while also not using the mediator's long term identifiable address. The buyer and seller can jointly generate the randomness and run this algorithm together so that they are both convinced that it was run properly.

To redeem the escrowed funds, the party to be paid will hand over $\hat{x}$ to the mediator. Using $(\hat{x})$ together with its secret key x, the mediator can now sign over the key y', and thus create the pay-out transaction.

Properties. This naive scheme has many drawbacks: it is not secure, not optimistic, and not internally hiding. On the other hand, the scheme is not active-on-deposit and its simplistic nature scheme is somewhat privacy-preserving as it is both dispute hiding and externally hiding.

5.2 Escrow via `Multisig`

A well-known improvement uses Bitcoin's multisig feature. In this scheme, the money is not sent directly to the escrow service's address, but instead it is sent to a 2-of-3 multisig address with one key controlled by each of the transacting parties and one controlled by the mediator. When there is no dispute, the two transacting parties can together create the pay-out transaction. Only when there is a dispute will the mediator get involved, collaborating with either the buyer or seller (as appropriate) to redeem the funds. This scheme is available today.[5]

As in the Silk Road scheme, for the sake of adding privacy, rather than including a longstanding address that is publicly associated with the mediator, the parties can use a blinded address.

Properties. This protocol is secure as the mediator cannot unilaterally redeem the escrowed funds. It is also optimistic. However it is susceptible to denial-of-service attack as the mediator can refuse to mediate a dispute.

The use of a blinded address for the mediator makes this scheme internally-hiding. The 2-of-3 structure makes the scheme not externally hiding, however, and it is also not dispute-hiding as one may be able to detect a dispute through transaction graph analysis (see Appendix C.2 for more details).

If one's goal is an escrow scheme that is transparent, then the non-blinded version of this scheme is a good candidate as it is secure and allows blockchain observers to detect disputed transactions.

5.3 Escrow via threshold signatures

Replacing the 2-of-3 multisignature address with a single 2-of-3 threshold address improves the privacy of this scheme. With threshold signatures, the three parties jointly generate *shares* of a regular single key address such that any 2 of them can jointly spend the money in that address. Unlike multisig, this threshold address is indistinguishable from a typical address and to an external observer would

[5] See for example https://escrowmybits.com/.

look like the (blinded) Silk Road scheme. Moreover, the signed transaction on the blockchain does not give any indication as to which parties participated in generating the signature.

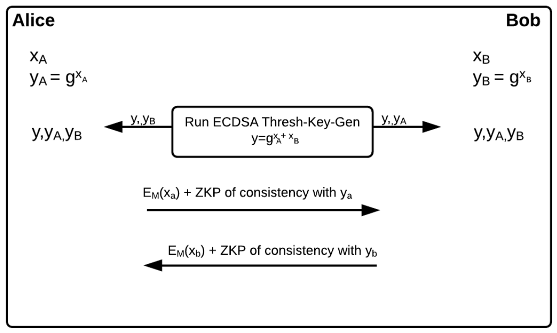
Properties. This scheme is secure, externally hiding, and dispute hiding. It is not, however, optimistic as it is active-on-deposit − the threshold signature scheme requires an interactive setup in which all 3 parties must participate. It is also susceptible to denial of service. It also generates a new key every time that both parties as well as the mediator must keep track of.

5.4 Escrow via encrypt-and-swap

We now present a new optimistic protocol which meets all of our privacy properties:

1. Alice and Bob generate a $2-of-2$ shared ECDSA key. Note that we do not need a full threshold scheme, but the `Thresh-Key-Gen` protocol of Gennaro al. [27] is suitable to generate the secret shares in a distributed manner. At the end of the protocol, Alice has x_A, Bob has x_B, and the shared public key is $y = g^{x_A+x_B}$. Moreover, as part of the protocol, both parties learn $y_A = g^{x_A}$ and $y_B = g^{x_B}$.
2. Alice sends $c_A = E_M(x_A)$, an encryption of her secret x_A under M's public key, to Bob together with a zero-knowledge proof[6] π_A that $g^{D(c_A)} = y_A$.
3. Bob sends $c_B = E_M(x_B)$ to Alice together with a zero-knowledge proof π_B that $g^{D(c_B)} = y_B$.
4. In the absence of a dispute, Alice sends x_A to Bob and Bob can now transfer the funds to his own account. Conversely, if both parties agree that a refund is in order, Bob sends x_B to Alice.
5. In the event of a dispute, the mediator investigates and chooses the "winner", which we'll denote $W \in \{A, B\}$. The winner sends c_W to M. M decrypts it and sends it back to W, who now has both shares of the key and thus can sign a redeem transaction.

[6] The zero knowledge proof proves that a ciphertext encrypts the discrete log of a known value for a known base. For details of how to construct this proof see Camenisch *et al.* [22]. Gennaro *et al.* demonstrates that these proofs work with ECDSA and Bitcoin keys [27].



The encrypt-and-swap escrow protocol

Equivocation This scheme introduces the risk of an *equivocation* attack in which a malicious mediator tells both parties that they won the dispute. Each party will give their ciphertext to the mediator, at which point the mediator can reconstruct the entire key and steal the escrowed funds.

We can prevent this attack by replacing 2-of-2 shared address with a 3-of-3 address. The third share x_C will be shared by the transacting parties and never given to the mediator. This way, even if the mediator equivocates, it will only receive two shares x_A and x_B and cannot transfer the money.

Properties. This protocol is both secure and optimistic. Moreover it satisfies all of our privacy properties: it is internally-hiding, externally-hiding, and dispute-hiding (on the blockchain, it appears as if funds were sent to an ordinary address). The only con of this scheme is that it is susceptible to a denial-of-service attack.

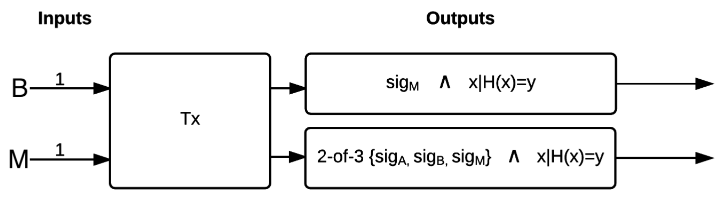
5.5 Escrow with bond

We now present a scheme that is resilient to denial-of-service attacks. To do this, we include an incentive system to punish a mediator who fails to release the funds from escrow. At a high level, we require the mediator to deposit a surety bond alongside the transacting parties. The general idea of preventing denial-of-service attacks by having the third party put money in bond has appeared in other contexts [31].

We make use of a feature of the Bitcoin scripting language that requires one to present an x such that SHA-256$(x) = y$ as a condition of spending money. This feature has previously been used, for example, to construct atomic cross-chain swap protocols (see Section 3.2) or offline micropayment channels [41].

We use this feature to build a transaction that ensures that the mediator will only be able to take his money out of bond if the escrow transaction is resolved.

1. Alice and Bob agree on a value x that is unknown to the mediator. They compute $y =$SHA-256(x).
2. Bob, the buyer, creates a transaction with two inputs. One input is the funds that he is putting in escrow; the other is the mediator's bond. The bond should be equal to Bob's payment amount.
3. The first output of the transaction requires 2-of-3 of Alice, Bob, and the mediator to sign. Moreover, it requires a SHA-256 preimage of y (e.g. x).
4. The second output requires a signature from the mediator as well as a SHA-256 preimage of y.



Bond Protocol to prevent denial-of-service attack

In the absence of a dispute, Alice and Bob can redeem the first output themselves). However, in the process, they must reveal x publicly on the blockchain, which the mediator can then use to recover their bond. In case of a dispute, Alice and Bob will refuse to reveal x until the mediator chooses a winner and signs the output transaction, preventing the mediator from recovering the bond until the dispute is resolved.
Properties. This protocol is both secure and resistant to denial of service. However, it is active-on-deposit and not externally-hiding, internally-hiding, nor dispute-hiding.

6 Group escrow

The escrow protocols we have described so far assume that there is a single mediator. Moreover, only one of these schemes (escrow bond) was resistant to denial-of-service attacks, and this scheme achieved this property at the expense of being active-on-deposit and compromising privacy.

Instead, we propose an entirely different way to deal with denial-of-service attacks (as well as improve resistance to collusion attacks or a mediator simply going offline). By distributing the signing power among n mediators who will resolve disputes by a majority vote (we assume n is odd), we can ensure that no single mediator has the ability to abort and deny service. As long as the majority are willing to complete the protocol, a denial-of-service attack is thwarted.

A recurring lesson in Bitcoin's history is that putting trust in any single party is risky. Bitcoin has been plagued by *exit scams* in which third-party services gain consumer trust and then disappear [5]. In a study of 40 Bitcoin exchanges, Moore and Christin find that approximately 40% of these services went under, often leaving no funds and no recourse for the customers that trusted them [39]. In 2014, the then-largest exchange, Mt. Gox, famously claimed to have lost 850,000 bitcoins, and passed the losses directly to its customers.

6.1 Definitions and models

Our definitions from single-mediator escrow protocols all remain in place, with the exception that security now requires protection from theft even if *all* of the mediators collude.

Definition 6.1 (Secure group escrow protocol). *A group escrow service is said to be secure if an external adversary that fully controls all of the mediators cannot transfer any of the money being held in escrow.*

We discuss two different models for how such groups of escrow services are assembled. First, we might use an *ad-hoc group* of mediators. In this model, the buyer and seller are free to (jointly) choose anybody with a Bitcoin address to serve as mediators and the mediators need not ever communicate with each other. Note that only one mediator must be jointly chosen (and trusted). The buyer may choose k mediators and the seller chooses k mediators. They then jointly choose 1 mediator as a "tie-breaker."

We can also leverage *predetermined groups* which have already communicated and agreed to work together. The buyer and the seller merely choose one of these groups to act as their mediator service.

6.2 Group escrow via `Multisig`

We can build a scheme using a script specifying that the funds can be redeemed if either (1) the transacting parties both sign or (2) one of the transacting parties together with a majority of the mediators signs. Using A and B to represent signatures by the transacting parties and $M_1, \ldots, M_{2n+1}$ to represent respective signatures by the mediators, the script will check that the following Boolean formula is satisfied:

$$(A \wedge B) \vee (A \wedge \text{n+1-of-}\{M_1, \ldots, M_{2n+1}\}) \vee (B \wedge \text{n+1-of-}\{M_1, \ldots, M_{2n+1}\})$$

For privacy, the mediators' addresses can be blinded as before.

While Bitcoin does limit the number of signature operations that a script can contain, the limits are reasonable in practice. In particular, using Bitcoin's pay to script hash (P2SH) feature, one can create a script that specifies 15 signature operations [8]. The script above requires up to $4 + 2m$ signature operations to validate, where m is the number of moderators. Thus, this script would be acceptable for $m \leq 5$, and in practice 5 mediators will generally be sufficient.

Properties. This scheme is secure and optimistic. Since the mediators' addresses are blinded, it is also internally hiding. However, it is neither dispute hiding nor internally-hiding. It is partially resistant to denial-of-service attacks as in order to launch such an attacks, the majority of the mediators must participate.

6.3 Group escrow via encrypt-and-swap

We can build a group-analog to our encrypt-and-swap scheme. As before, the transacting parties run the `Thresh-Key-Gen` protocol of [27] to generate a shared 2-of-2 threshold address. Once they run this protocol, Alice has her key share x_A, and Bob has x_B. Moreover, as a side effect of the `Thresh-Key-Gen` protocol, both parties learn $y_A = g^{x_A}$ and $y_B = g^{x_B}$

The parties then create a Shamir secret sharing of x_A and x_B. If there are $n = 2t + 1$ mediators, the transacting parties share their secret on a degree t polynomial, thus ensuring that a majority of the mediators is necessary and sufficient to recover the secret.

Using each mediator's public key M_i, each party encrypts the corresponding share to that mediator and gives all of these ciphertexts to the other party.

If there is no dispute, the party that is paying will give its key share to the other party, who now has both shares and can redeem the money.

In the event of a dispute, the mediators will vote. The winning party will give each mediator the corresponding ciphertext that it received from the other party. The mediators decrypt their shares, and a majority reconstructs the losing party's threshold key share. They then give this reconstructed key share to the winning party who can now create a pay-out transaction to itself.

If all players honestly follow the protocol, it is clear that this protocol is both secure and correct. However, we wish to achieve security against a malicious player that may deviate from the protocol. Intuitively, in order to achieve this, each party needs to prove to the other party that the values that it gives it are indeed Shamir-secret sharings of their threshold secret share.

We implement this proof in two phases: for each mediator, when Alice gives Bob the ciphertext $c_i = E_{M_i}(P_i)$, Alice additionally includes a Feldman VSS (see Appendix A.3) value $w_i = g^{P_i}$ as well as a zero-knowledge proof of consistency between these two values. Using Feldman's scheme, Bob then verifies that w_i is indeed a Shamir secret-share of x_A.

We now present the details of this protocol:

1. Alice and Bob run `Thresh-Key-Gen` of Gennaro *et al.* [27] to generate a shared 2-of-2 ECDSA key. Alice has x_A, Bob has x_B, and the shared public key is $y = g^{x_A + x_B}$. As part of the protocol, both parties learn $y_A = g^{x_A}$ and $y_B = g^{x_B}$.
2. Alice shares x_A over a degree t polynomial with coefficients $a_1, \ldots, a_t$.

$$P^{(a)}(w) = x_A + a_1 w + \cdots + a_t w^t$$

3. Alice computes a share $P_i^{(a)} = P^{(a)}(i)$ for each mediator and encrypts that mediator's share under their public key as follows:

$$c_i^{(a)} = E_{M_i}(P_i^{(a)})$$

   Alice gives Bob $\{c_1, c_2, \ldots, c_n\}$.

4. For each mediator's share, Alice computes $w_i^{(a)} = g^{P_i^{(a)}}$ and gives Bob $\{w_1^{(a)}, w_2^{(a)}, \ldots, w_n^{(a)}\}$.
5. For each mediator, Alice gives Bob a zero knowledge proof $\Pi_i^{(a)}$ that states

$$g^{D_{M_i}(c_i^{(a)})} = w_i^{(a)}$$

   That is Alice proves that the $c_i^{(a)}$ is an encryption of the discrete log with respect to g of $w_i^{(a)}$.
6. Alice creates a Feldman VSS of the shared secret. In particular, she gives Bob $c_1^{(a)} = g^{a_1}, \ldots, c_n^{(a)} = g^{a_n}$. Bob already has $c_0^{(a)} = g^{x_A}$ as this was output in step 1.
7. Bob verifies each of the zero-knowledge proofs $\Pi_i^{(a)}$. Bob also verifies $\forall i$,

$$w_i^{(a)} = c_0^{(a)} \cdot (c_1^{(a)})^i \cdot (c_2^{(a)})^{i^2} \cdots (c_t^{(a)})^{i^t}$$

   If any of these checks fail, Bob aborts.
8. Bob and Alice perform steps 2-8 in reverse.
9. Now that each party is convinced that they hold the VSS of the other party's share encrypted to the mediators, Bob (the buyer) deposits the money in the escrow address.

Properties. This protocol is secure, optimistic, internally-hiding, externally-hiding, and dispute-hiding. It supports ad-hoc groups. Moreover, it's group nature means that it has partial denial-of-service resistance as in order to launch such an attack, a majority of the mediators must participate.

		Not active-on-deposit	Not active-on-withdrawal	Optimistic	Secure	DoS resistant	Dispute-hiding	Externally-hiding	Internally-hiding	Works for predetermined groups	Works for ad-hoc groups
Protocol	*Sec.*	*Activity*			*Security*		*Privacy*			*Groups*	
Direct payment	5.1						●	●			
2-of-3 multisig	5.2	●	●	●	●		○	○	●		
2-of-3 threshold signature	5.3		●		●		●	●			
Encrypt-and-swap	5.4	●	●	●	●		●	●	●		
Escrow with bond	5.5	●			●	●		○			
Group multisig	6.2	●	●	●	●	○			●	●	●
Group encrypt-and-swap	6.3	●	●	●	●	○	●	●	●	●	●

● fully achieves
○ partially achieves

Table 1. Comparative evaluation of escrow schemes.

7 Conclusion

We have proposed a number of protocols, as summarized and compared in Table 1. Assuming the goal is complete privacy, our recommendation is to use group escrow via encrypt-and-swap as it comes closest to fulfilling all of the properties that we set forth. If, however, the goal is transparency, then one should choose the non-blinded version of the 2-of-3 multisig scheme or the group multisig scheme as they are not dispute-hiding.

8 Acknowledgements

We would like to thank Andrew Miler and Washington Sanchez for useful discussions and feedback.

Steven Goldfeder is supported by the NSF Graduate Research Fellowship under grant number DGE 1148900. Rosario Gennaro is supported by NSF Grant 1545759. Arvind Narayanan is supported by NSF Grant CNS-1421689.

References

1. Bitcoin wiki: Atomic cross-chain trading. https://en.bitcoin.it/wiki/Atomic˙cross-chain˙trading , accessed: 2016-11-14
2. Bitcoin wiki: Elliptic Curve Digital Signature Algorithm. https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm, accessed: 2014-02-11
3. Bitcoin wiki: Secp265k1. https://en.bitcoin.it/wiki/Secp256k1, accessed: 2016-11-01
4. Bitcoin wiki: Transactions. https://en.bitcoin.it/wiki/Transactions, accessed: 2016-11-01
5. Monero Loses Darknet Market in Apparent Exit Scam. https://cointelegraph.com/news/monero-loses-darknet-market-in-apparent-exit-scam , accessed: 2016-11-14
6. Stealth payments. http://sx.dyne.org/stealth.html, accessed: 2016-11-14
7. Open bazaar protocol. docs.openbazaar.org (2016)
8. Andresen, G.: Github: Proposal: open up IsStandard for P2SH transactions. https://gist.github.com/gavinandresen/88be40c141bc67acb247, accessed: 2017-02-16
9. Andresen, G.: Github: Shared Wallets Design. https://gist.github.com/gavinandresen/4039433, accessed: 2016-11-01
10. Andrew, M.: Bitcoin forum post: Alt chains and atomic transfers
11. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł.: Fair two-party computations via bitcoin deposits. In: International Conference on Financial Cryptography and Data Security. pp. 105–121. Springer (2014)
12. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 443–458. IEEE (2014)
13. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: Proceedings of the 4th ACM conference on Computer and communications security. pp. 7–17. ACM (1997)

14. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Advances in CryptologyEUROCRYPT'98, pp. 591–606. Springer (1998)
15. Bahreman, A., Tygar, J.: Certified electronic mail. Master's thesis, Carnegie Mellon University (1992)
16. Banasik, W., Dziembowski, S., Malinowski, D.: Efficient zero-knowledge contingent payments in cryptocurrencies without scripts
17. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on. pp. 77–85. IEEE (1998)
18. Bentov, I., Kumaresan, R.: How to use bitcoin to design fair protocols. In: International Cryptology Conference. pp. 421–439. Springer (2014)
19. Blum, M.: Three applications of the oblivious transfer: Part i: Coin flipping by telephone; part ii: How to exchange secrets; part iii: How to send certified electronic mail. University of California, Berkeley, CA (1981)
20. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 104–121. IEEE (2015)
21. Cachin, C., Camenisch, J.: Optimistic fair secure computation. In: Annual International Cryptology Conference. pp. 93–111. Springer (2000)
22. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Advances in Cryptology-CRYPTO 2003, pp. 126–144. Springer (2003)
23. Christin, N.: Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In: Proceedings of the 22nd international conference on World Wide Web. pp. 213–224. International World Wide Web Conferences Steering Committee (2013)
24. Danezis, G., Meiklejohn, S.: Centrally banked cryptocurrencies. arXiv preprint arXiv:1505.06895 (2015)
25. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Foundations of Computer Science, 1987., 28th Annual Symposium on. pp. 427–438. IEEE (1987)
26. Garay, J.A., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: Annual International Cryptology Conference. pp. 449–466. Springer (1999)
27. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security
28. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold dss signatures. In: Advances in CryptologyEUROCRYPT96. pp. 354–371. Springer (1996)
29. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Advances in CryptologyEUROCRYPT99. pp. 295–310. Springer (1999)
30. Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version (1998)
31. Jakobsson, M.: Ripping coins for a fair exchange. In: Advances in CryptologyEUROCRYPT95. pp. 220–230. Springer (1995)
32. Juels, A., Kosba, A., Shi, E.: The ring of gyges: Using smart contracts for crime. aries 40, 54 (2015)
33. Küpçü, A., Lysyanskaya, A.: Usable optimistic fair exchange. In: Cryptographers Track at the RSA Conference. pp. 252–267. Springer (2010)
34. Lindell, A.Y.: Legally-enforceable fairness in secure two-party computation. In: Topics in Cryptology–CT-RSA 2008, pp. 121–137. Springer (2008)

35. MacKenzie, P., Reiter, M.K.: Two-party generation of dsa signatures. International Journal of Information Security 2(3-4), 218–239 (2004)
36. Maxwell, G.: The first successful zero-knowledge contingent payment
37. Maxwell, G.: Zero knowledge contingent payment
38. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: Proceedings of the twenty-second annual symposium on Principles of distributed computing. pp. 12–19
39. Moore, T., Christin, N.: Beware the middleman: Empirical analysis of bitcoin-exchange risk. In: Financial cryptography and data security, pp. 25–33. Springer (2013)
40. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted 1, 2012 (2008)
41. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments. Tech. rep.
42. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
43. Wuille, P.: Bip 32 : Hierarchical deterministic wallets. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, accessed: 2016-11-14

A Additional Background

A.1 Bitcoin

Bitcoin is a decentralized digital currency first proposed in 2008 [40]. We present a simplified treatment here; we refer the reader to [20] for detailed survey.

Bitcoins can be thought of as being owned by *addresses*; an address is simply the hash of a public key. To transfer bitcoins from one address to another, a *transaction* is constructed that specifies one or more input addresses from which the funds are to be debited, and one or more output addresses to which the funds are to be credited. For each input address, the transaction contains a reference to a previous transaction which contained this address as an output address. In order for the transaction to be valid, it must be signed by the private key associated with each input address, and the funds in the referenced transactions must not have already been spent [40, 4].

Signed transactions are broadcast to the Bitcoin peer-to-peer network. They are validated by *miners* who group transactions together into *blocks*. Miners participate in a distributed consensus protocol that collects these blocks into an append-only global log called the *blockchain*.

Our treatment of transactions thus far has described what a *typical* Bitcoin transaction looks like. However, Bitcoin allows for far more complex transactions. Bitcoin in fact specifies a *script* written in a stack-based programming language which defines the conditions under which a transaction may be redeemed. Every transaction contains a *script* that specifies how the transferred funds may be redeemed. For a typical transaction, the script specifies a single address which must sign any redeeming transaction to move the funds. This scripting language also include support for *multisignature* scripts [9] which require at least t of n specified public keys to provide a signature on the redeeming transaction.

A.2 ECDSA

We use generic G-DSA signature notation from [27]. The public parameters include a cyclic group $\mathcal{G}$ of prime order q generated by an element g, a hash function H defined from arbitrary strings into Z_q, and another hash function H' defined from $\mathcal{G}$ to Z_q. G-DSA consists of three algorithms:

- Key Generation A secret key x is chosen uniformly at random in Z_q. The corresponding public key $y = g^x$ is computed in $\mathcal{G}$.
- Signing on input an arbitrary message M, we compute $m = H(M) \in Z_q$. Then the signer chooses k uniformly at random in Z_q and computes $R = g^k$ in $\mathcal{G}$ and $r = H'(R) \in Z_q$. Then she computes $s = k^{-1}(m + xr) \bmod q$. The signature on M is the pair (r, s).
- Verification On input $M, (r, s)$ and y, the receiver checks that $r, s \in Z_q$ and computes
$$R' = g^{ms^{-1} \bmod q} y^{rs^{-1} \bmod q} \text{ in } \mathcal{G}$$
and accepts if $H'(r') = r$.

The traditional DSA algorithm is obtained by choosing large primes p, q such that $q|(p-1)$ and setting $\mathcal{G}$ to be the subgroup of Z_p^* of order q. In this case the multiplication operation in $\mathcal{G}$ is multiplication modulo p. The function H' is defined as $H'(R) = R \bmod q$.

The ECDSA scheme is obtained by choosing $\mathcal{G}$ as a group of points on an elliptic curve of cardinality q. In this case the multiplication operation in $\mathcal{G}$ is the group operation over the curve. The function H' is defined as $H'(R) = R_x \bmod q$ where R_x is the x-coordinate of the point R.

A.3 Feldman Verifiable Secret Sharing

A *verifiable* secret-sharing (VSS) scheme is one in which the recipients of secret shares can verify that they have received consistent shares. In [25], Feldman extended Shamir's secret sharing scheme to build a VSS scheme.

Feldman's scheme is defined over or a group G with generator g for which computing discrete logarithms is hard. To verifiably share a secret x in such a group such that $t+1$ players can reconstruct it, the dealer first chooses a random polynomial as in Shamir's scheme:

$$P(w) = s + a_1 w + \cdots + a_t w^t$$

The ith player is given a share $P_i = P(i)$, and $t+1$ players can reconstruct the polynomial via Lagrange interpolation as in Shamir's scheme.

Next, the dealer also publishes $c_0 = g^s, c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$. Now in order to verify that their shares are consistent, each player computes the polynomial in the exponent and verifies that:

$$g^{P_i} = c_0 \cdot (c_1)^i \cdot (c_2)^{i^2} \cdots (c_t)^{i^t}$$

B ECDSA blinding security argument

Privacy. Because y' is uniformly distributed in Z_q, the public keys y and y' are not linkable.
Security.

We can easily show that the new public key y' is unforgeable given $\hat{x}$ without knowledge of the original private key x.

Assume that given $(y, y, \hat{x}, \hat{y})$ that algorithm A could forge a signature on y. Then we can build a general forger F for ECDSA as follows: When given a key y to forge one can do as follows:

- choose random $\hat{x}$
- compute $\hat{y} = g^{\hat{x}}$
- compute $y = \frac{y}{\hat{y}}$
- Run A on input $(y, y, \hat{x}, \hat{y})$ and return whatever A outputs

The tuple $(y, y, \hat{x}, \hat{y})$ has the same distribution as the one we are trying to forge. Thus our blinding is secure by reduction to ECDSA security.

C Detailed analysis of escrow schemes

C.1 Detailed analysis of escrow via direct payment

Not secure. This service does not meet our definition of security. In particular, an adversary who gains control of the escrow service can simply transfer all of the funds held in escrow to his own address.

This service is clearly active on withdrawal as the mediator is in full control of the funds and must send the funds to the appropriate party even if there is no dispute.
Not active-on-deposit. This scheme is not active-on-deposit as parties can send money to the mediator without its active participation.

Indeed, the above scheme could be simplified and instead of using blinding to achieve privacy, the mediator generates a new address for each escrow transaction. This, however, would make the scheme active-on-deposit, and we use blinding to gain privacy while remaining not active-on deposit.
Active-on-withdrawal. This scheme is active-on-withdrawal as even if there is no dispute, the escrow service needs to release the funds from its own address.
Dispute-hiding. This scheme is dispute-hiding. In all cases whether or not there was a dispute, the escrow service will forward the money to a single address, and thus this does not reveal whether or not there was a dispute. [7]

[7] Of course, if there was a dispute and the buyer won, the money would be sent back to the buyer's address. But assuming that the buyer provides the escrow service with a return address that is not linked to his original address, then nothing is learned here. It is possible that someone could correlate the buyer's various addresses, but this is not a weakness in our escrow system as the system does not change when there is a dispute.

Externally-hiding. The escrow and pay-out transactions both only contain regular pay to single-key transactions. Moreover, since the mediator's address is blinded, an external observer will not be able to detect that this address belongs to a mediator. Indeed the purpose of using a blinded address is to achieve this property.
Not internally-hiding. Although the address is blinded, since the scheme is active-on-withdrawal, it is not internally-hiding. Even if there is no dispute, the transacting parties must inform the mediator that their service is being used so that the service can release the funds.

It is true that the mediator will not be able to detect that its being used without the blinding secret. And the parties can delay telling the service the secret until they want to take the money out of escrow. But nevertheless, they must in all cases involve the mediator, and thus it is not internally-hiding.
Susceptible to denial-of-service attack. This scheme does not protect the transacting parties against denial-of-service attack. The escrow agent could refuse to respond. Importantly, since this scheme is active-on-withdrawal, the mediator could deny service even when there is no dispute between the transacting parties.

C.2 Detailed analysis of escrow via `Multisig`

Secure. An external attacker will only learn the single key controlled by the escrow service. Yet in order to move funds, it must also obtain a second key belonging to one of the transacting parties. Since we refer to an external attacker who has not compromised one of the transacting parties, there is no way for the attacker to create a pay-out transaction, and this scheme therefore meets our definition of security.
Not active-on-deposit. This scheme is not active-on-deposit as parties can send money to the mediator without its active participation.
Not active-on-withdrawal. When the transacting parties are in agreement, they can together transfer the money to the correct party without the involvement of the escrow service. This is due to the fact that the transacting parties together control two keys, and the transaction only requires two of the three keys in order to create a pay-out transaction.
Optimistic Since the scheme is neither active-on-deposit nor active-on-withdrawal, it fulfills the definition of an optimistic mediator.
Partially Dispute-hiding. This scheme is somewhat dispute-hiding. If an outsider cannot determine which addresses belong to which party, then it is dispute hiding. Yet, an outsider may be able to do transaction graph analysis on the multisignature keys to determine which is controlled by the buyer and seller and thus learn whether or not a dispute occurred.

To combat this problem, the buyer and seller, should use fresh keys in this transaction, and never use them again. However, it is hard to gain any provable privacy here as transaction graph analysis may be able to recover information.

In particular, if there was a dispute and the seller won, the resulting transaction would be identical to the buyer awarding payment to the seller when there was no dispute.

As an illustration as to why we cannot guarantee that this scheme is dispute-hiding consider the following scenario: There is a dispute in the transaction. The winner then goes ahead and combines the money that he was just paid with some other money that he has to purchase something. Transaction graph analysis can link the winning address to the seller.

Now, as long as the other two address in the multisig transaction are never used again, this transaction will still be dispute hiding. Since these two keys only appeared in the blockchain once, it will be impossible to determine which belongs to the losing party and which belongs to the escrow agent, and thus it will be impossible to determine whether the losing party signed (i.e. there was no dispute) or the mediator signed (i.e. there was a dispute).

But, there is nothing stopping the losing party from going ahead and using his key further. That is, we cannot guarantee that she will never use his key again and linking it to her identity. Indeed if she does this, it will become immediately clear that her key was not the second key used to create the pay-out transaction, and thus it is clear that a dispute occurred and the mediator was involved.

Contrast this to the Silk Road scheme which is fully externally-hiding. In the Silk Road scheme, nothing on the blockchain changes at all whether or not there was a dispute. In either case the mediator will create pay-out transaction sending the money to the winning party. Transaction graph analysis would thus be useless.

Partially Externally-hiding. The scheme is somewhat externally hiding. As the blinded service's address is not linked to any of its other addresses, an outsider will be unable to know that escrow is being used. Yet, since it's a 2-of-3 multisig address and these addresses are often used for escrow, there is some information being leaked, and thus it is less private than a scheme that uses a standard single-key address.

Internally-hiding. This scheme is internally-hiding since the mediator cannot link its key y to the blinded key y' since y' is uniformly distributed independent of y. The mediator would thus be unable to tell that its service has been used unless one of the parties contacts them and provides them with the blinding secret. Since this will only happen when there is a dispute, the scheme is internally-hiding.

Note that this is an example of a scheme that is only partially externally-hiding, yet it is internally-hiding. The reason is that although there may be some leaked information allowing one to determine that escrow is being used, there is no leak that would allow anyone–including the mediator itself–to determine who the mediator is.

Susceptible to denial-of-service attack. This scheme does not protect the transacting parties against denial-of-service attack. The escrow agent could refuse to respond. However, since this scheme is not active-on-withdrawal, the mediator can only deny service in case of a dispute. When there is no dispute, the transacting parties can take the money out of escrow by themselves.

C.3 Detailed analysis of escrow via threshold signatures

Secure. As with the multisignature scheme, the 2-of-3 structure makes this a secure scheme. An external attacker that compromised the mediator cannot create a pay-out transaction.
Active-on-deposit. This scheme is active-on-deposit as the threshold signature address generation scheme requires the active participation of all involved parties. Indeed, not only is it active on deposit, the process of generating a threshold shared key is computationally expensive.

However, it is only active on *first* deposit. If these same two parties wish to transact multiple times with the same mediator, they can do so using (a blinded version of) the same address, and thus do not need to repeat the key generation.
Not active-on-withdrawal. It is not active on withdrawal as the two transacting parties can themselves generate a pay-out transaction when there is no dispute.
Dispute-hiding. This scheme is dispute-hiding since the signature and the transaction that spends money from this address does not leak any information on which of the three parties participated.
Externally-hiding. This scheme is externally-hiding as the threshold signature key-generation protocol generates an address that is in no way associated with any of the mediator's addresses. Moreover, the transaction looks like a regular single key transactions, and there is no indicator on the blockchain that escrow is being used.
Not internally-hiding. Since the scheme is active-on-deposit, it is not internally hiding.
Susceptible to denial-of-service attack. This scheme is susceptible to a denial-of-service attack. However, as with the multisignature scheme, because it is non active-on-withdrawal, this attack can only be launched when there is a dispute.

C.4 Detailed analysis of escrow via encrypt-and-swap

Secure. This scheme is secure. An external attacker who corrupts the mediator will learn nothing as the mediator will have at most one share of the DSA key whereas two are needed to sign.
Not active-on-deposit. The third party has no role at all during the deposit phase. Although Alice and Bob encrypt using the third parties keys, they deposit these ciphertexts with one another and not with the third party.
Not active-on-withdrawal. When there is no dispute, Alice and Bob can threshold sign the transaction without involving the third party.
Optimistic Since the scheme is neither active-on-deposit nor active-on-withdrawal, it fulfills the definition of an optimistic mediator.
Dispute-hiding. This scheme is dispute hiding as on the blockchain it is simply a single signature address and the signature leaks no information about which parties participated in signing.

Externally-hiding. This scheme is externally hiding as it is just a single signature address on the blockchain and there is no indication that escrow is being used.
Internally-hiding. Alice and Bob generate a new address that is not at all related to the mediator's key. Thus there is no way for the mediator to detect that its service is being used from examining the blockchain.
Susceptible to denial-of-service attack. The only property that this scheme lacks is that it is susceptible to a denial of service attack. As before, since it is not active-on-withdrawal, this attack is limited to cases where there is a dispute.

C.5 Detailed analysis of escrow with bond

Secure. This scheme is secure as it uses 2-of-3 multisig, and an external attacker that corrupts the mediator cannot transfer the money.
Active-on-deposit. The mediator must be involved at the deposit phase, and thus this scheme is active-on-deposit. We note that the mediator's involvement is inherent to the scheme, and it is an open question whether we can achieve a scheme that is non-active-on-deposit and also not vulnerable to denial-of-service attacks.
Not active-on-withdrawal. If the parties agree, they can create the redeem transaction by themselves. The winning party will get paid, and in the process, they will reveal x so the mediator will also receive its funds.
Dispute-hiding. The scheme is somewhat dispute hiding –just as is the multisignature scheme. ***Not Externally-hiding.*** This scheme has an obvious structure as it uses a non-standard transaction, and it is thus not externally hiding.
Not internally-hiding. Since this scheme is active-on-deposit, it is inherently not internally-hiding.
Not susceptible to denial-of-service attack. The bond mechanism leaves the mediator with an incentive to resolve the dispute thus protecting against a denial-of-service attack.

Detailed analysis of group escrow via `Multisig`. ***Secure.*** This scheme is secure. The script directly enforces that only Alice and Bob or one of them together with a majority of the mediators can create a pay-out transaction. Thus, even if an attacker has compromised all of the mediators, he can still do nothing without either Alice or Bob.
Not active-on-deposit. This scheme is not active on deposit.
Not active-on-withdrawal. The first clause of the script allows Alice and Bob alone to redeem, so it is not active-on withdrawal.
Optimistic Since the scheme is neither active-on-deposit nor active-on-withdrawal, it fulfills the definition of an optimistic mediator.
Not dispute-hiding. This scheme is not dispute hiding at all. In case of dispute one of the second two clauses will be fulfilled, whereas when there is no dispute, the first clause will be fulfilled. Since these clauses are syntactically different, it becomes clear from examining the blockchain whether or not there was a dispute.

Not externally-hiding. This transaction has a strange script that will immediately signal to any observer that escrow is being used.
Internally-hiding. It is internally-hiding since the mediator's addresses are blinded so they cannot identify that their service is being used unless there is a dispute that they are asked to resolve.
Partially resistant to denial-of-service attack. This group nature of this scheme means that it will only be vulnerable to a denial of service attack if the majority of mediators abort, and thus it does have some built in protection against denial of service attacks.

Detailed analysis of group escrow via encrypt-and-swap ***Secure.*** Since $c_0 = g^{x_A}$, Bob knows that the polynomial is indeed sharing of Alice's secret x_A (and vice versa, Alice knows the same for Bob and x_B). Now, Bob just needs to verify that each of the encrypted shares are actually shares of this polynomial. For each mediator's share, Bob verifies that

$$w_i = c_0 \cdot (c_1)^i \cdot (c_2)^{i^2} \cdots (c_t)^{i^t}$$

Bob is thus convinced that each $\log_g(w_i)$ is a shamir-share of Alice's secret x_A. Moreover, the zero knowledge proofs prove to bob that $c_i = \log_g(w_i)$, and thus putting these both together, Bob is convinced that each ciphertext encrypts a share of x_A.

The security of the scheme thus follows from the security of the secret sharing x_A and x_B. For the Feldman scheme, since Alice publishes the Feldman commitments (i.e. the w_i's), we require that these are computed in a group where finding discrete logarithms is hard. The `Thresh-Key-Gen` scheme itself requires this same property as well.
Not active-on-deposit. This scheme is not active on deposit as the moderators have no role unless there is a dispute.
Not active-on-withdrawal. Alice and Bob can jointly threshold-sign if there is no dispute, so this scheme is not active-on withdrawal.
Optimistic Since the scheme is neither active-on-deposit nor active-on-withdrawal, it fulfills the definition of an optimistic mediator.
Dispute-hiding. This scheme is dispute-hiding as no matter what happens, the redeem transaction is just an ordinary transaction on the blockchain. Indeed, the signature will be the same whether the moderator was involved or the party's agreed and settled without the moderator's involvement.
Externally-hiding. This protocol is externally-hiding since on the blockchain, it appears only as an ordinary single-address transaction and thus leaks no information that it is an escrow transaction.
Internally-hiding. It is internally-hiding since on the blockchain, it appears only as an ordinary single-address transaction. Moreover, the mediators have no involvement unless there is a dispute. Although each party uses the mediators' keys to encrypt shares during the setup phase, they do not give these to the mediators, but instead hand these ciphertexts to the other party. Only in case of dispute do the mediators receive these shares.

Partially resistant to denial-of-service attack. This group nature of this scheme means that it will only be vulnerable to a denial of service attack if the majority of mediators abort, and thus it does have some built in protection against denial of service attacks.

D Paying the mediator

An important question that is very related to the privacy properties that we discuss is the business model of the escrow service, and how it is to be paid.

If an escrow service is active-on-deposit, then it can negotiate its fees and receive payments for its services upfront. If, on the other hand, it is not active-on-deposit, and especially if it is internally hiding (so that the escrow service cannot even detect that money has been put in escrow with it), then a different business model is needed.

In this case, the escrow service will charge for dispute resolution. It is important that the escrow service has a clear fee schedule that determines how much it is to be paid for resolving a dispute. It can perhaps charge based on time required to investigate.