ChainArgos.com

Better blockchain intelligence.

OPINION

Blockchain Crimes
With Nobody to Charge?

WHAT BLOCKCHAIN RECORDS CAN AND CANNOT PROVE ABOUT CRIMINAL CONSPIRACY

NOVEMBER 17, 2026



Contents

EXECUTIVE SUMMARY 3

1. SCOPE OF THIS OPINION PIECE 5

2. CONSPIRACY BASICS 5

3. HOW A SAFE MULTISIG WORKS 7

3.1 Two further ways a Safe can act 12

4. SIGNING WITHOUT AN OWNER TRANSACTION 12

4.1 Recording approval on the blockchain 12

4.2 Approving away from the blockchain 14

Step One: Create an Owner Key 14

Step Two: Add the Newly-Created Blockchain Address to the Safe 15

Step Three: Specify the Action to Approve 15

Step Four: Sign the Approval 15

Step Five: Pass the Signatures to a Submitter 15

Step Six: Submit the Approved Action 16

The gas fee and the Safe's refund 16

4.3 What publishing a signature permits 17

4.4 Passing the signature in public 18

4.5 What is left to trace 20

5. THE ANTI-SHAGGY DEFENSE 22

6. PRACTICAL RESULTS 24

7. WHERE THE LAW AND POLICY STAND TODAY 26

7.1 The Tornado Cash prosecution 26

7.2 Department of Justice policy 27

7.3 Legislation 28

8. CONCLUSION 30

Appendix. Two further ways a Safe can act 31

info@chainargos.com

EXECUTIVE SUMMARY

This opinion piece seeks to answer whether a group of people who run a system on a blockchain can be held criminally liable if that system facilitates criminal activity. For instance, if one person sends a transaction that upgrades or changes that system, does the mere act of sending that transaction make the sender a participant in a criminal conspiracy?

A few terms need defining before the question can be answered.

A blockchain is a public ledger that anyone can read. This opinion piece assumes that no one can rewrite a blockchain entry once it has been settled. A transaction is an instruction sent to the blockchain and recorded in that ledger. A smart contract is a program stored on the blockchain that a transaction can cause to run. A deployment transaction, as this opinion piece uses the term, changes how a smart contract system works. "Permissionless" means that no gatekeeper decides who may send a transaction, so anyone may send one.[1] A crypto-asset is a unit of value recorded on a blockchain. On Ethereum, the blockchain considered here, sending a transaction incurs a gas fee. The gas fee depends on the work the transaction uses, measured in units called gas, and the price of each unit. This opinion piece uses "gas fee" for the charge and "gas" for the amount of work. A blockchain address is the public identifier of a party on the blockchain, and Section 3 sets out where one comes from. A multisig, short for "multiple signatures," is a smart contract that a group controls together. The smart contract acts only when enough of the group has approved a particular transaction. The Safe is the multisig product this opinion piece examines, and Section 3 sets out how it works. A Safe transaction is an action that the group approves for the Safe to carry out. A submitter is whoever sends the transaction that makes the Safe carry out that transaction. Keeping approval and submission separate is central to the argument.[2]

A criminal conspiracy under United States federal law is an agreement between two or more people to commit a crime.[3] An overt act in furtherance of such a criminal conspiracy is a step that one member of the group takes to carry out the plan.[4] The government must also prove that the defendant joined the conspiracy knowingly and intended to further its criminal object. The general federal conspiracy statute requires an overt act, although a number of other conspiracy statutes do not, and Section 2 explains the difference.[5]

Our view is that the sender of a transaction that upgrades or changes a system deployed on a blockchain could count as a conspirator to criminal activity under current US law.

Where the statute requires an overt act, sending a deployment transaction can satisfy that requirement if it furthers the criminal agreement. Where the statute requires no overt act, sending the

---

[1] Permissionless does not mean free of charge. Transaction fees may still apply.

[2] The definitions in this paragraph state the model of a blockchain that this opinion piece assumes, and the short descriptions of a multisig and of the Safe state what this opinion piece means by those terms. Those definitions and descriptions are the opinion piece's own rather than a claim about every blockchain or every product. The Safe contract source and the Safe documentation on which Sections 3 and 4 rest are cited from footnote 25 onward.

[3] 18 U.S.C. § 371, https://www.govinfo.gov/link/uscode/18/371. The statute reads in relevant part that "[i]f two or more persons conspire either to commit any offense against the United States, or to defraud the United States, or any agency thereof in any manner or for any purpose, and one or more of such persons do any act to effect the object of the conspiracy, each shall be fined under this title or imprisoned not more than five years, or both."

[4] *Yates v. United States*, 354 U.S. 298, 334 (1957) ("The function of the overt act in a conspiracy prosecution is simply to manifest 'that the conspiracy is at work,' . . . ."). The words "that the conspiracy is at work" are quoted by the Court from *Carlson v. United States*, 187 F.2d 366, 370 (10th Cir. 1951). For the knowledge and intent elements see Third Circuit Model Criminal Jury Instruction 6.18.371A and the accompanying elements and mental state provisions, in the chapter revised June 2026. The citation as the Court gives it supplies the volume, the first page, and the pinpoint page, and no circuit designation. A scan of the Federal Reporter page for *Carlson* is published by the Harvard Law School Library at https://static.case.law/f2d/187/case-pdfs/0366-01.pdf, with the case metadata at https://static.case.law/f2d/187/cases/0366-01.json. Neither file could be retrieved from the environment in which this opinion piece was prepared. The opinion was read instead in the reproduction published by OpenJurist at https://openjurist.org/187/f2d/366, which carries star pagination, on November 10, 2026. Printed page 370 reads "The overt act merely manifests that the conspiracy is at work." The sentence before it reads "Although innocent in itself, an overt act is sufficient to complete the crime of conspiracy if performed in furtherance thereof." The panel was Huxman, Murrah, and Pickett, Circuit Judges, and Judge Pickett concurred in part and dissented in part. The independent audit of this opinion piece retrieved the Harvard Law School Library scan separately and reports the same sentence on the same printed page, and confirms the Tenth Circuit designation printed at page 366.

[5] For the knowledge and intent elements see footnote 4. The text of 18 U.S.C. § 371 is at footnote 3.

deployment transaction is direct evidence that the sender caused the system to change. Both propositions concern the overt act and the evidence of agreement. Neither displaces the requirement that the sender joined the agreement knowingly and intended to further its criminal object. [6]

We reach that answer by proposing an evasion method, set out in Section 4, and then asking what the opposite view would leave the government able to charge in such situations. That opposite view holds that sending a deployment transaction is not in and of itself an overt act and does not by itself make the sender a conspirator. Adopting that view, against a group using the evasion method set out in this opinion piece, leaves no transaction on the blockchain record that the government can charge as the act advancing the criminal object.

The evasion method proposed rests on one feature of modern blockchain systems, which is that a group can vote on what the group will do without any member sending a transaction. Here, we are not referring to mixing services, offshore companies, or other ways of obscuring where crypto-assets came from. We mean that a vote can be cast without the voter sending a transaction of any kind, so that no transaction in the blockchain record is sent from a blockchain address the voter controls. The signatures the vote produces are published when the vote is executed, and the blockchain address that made each one can be recovered from it. Attributing a recovered blockchain address to a person, and establishing agreement, knowledge, and intent, require evidence well beyond the blockchain record, and Section 4.5 sets out what the published signatures do and do not show.[7]

The proposed evasion method uses the Safe's standard approval and submission process and requires no changes to the Safe. Despite appearances, the proposed evasion method is neither new nor innovative, because it simply operates the way the Safe is currently designed to work.[8]

Once it is understood how the proposed evasion method works, two readings of the deployment transaction remain open.

The first is that sending the transaction is not in and of itself an act advancing the criminal agreement, which leaves any charge against a group organized this way to rest on evidence other than the transaction that carried the signatures.

The second is to read current statutes on conspiracy law such that sending a permissionless deployment transaction is in and of itself an act advancing the criminal agreement. That reading applies where the sender joined the agreement knowingly and intended to further its criminal object. That reading keeps the deployment transaction available as evidence supporting a conspiracy charge.

Ultimately, only a judge and a jury can settle this question, but prosecutors can nonetheless adopt the "does not count" position without any court ruling. Prosecutors do so by declining to argue the point and by not prosecuting the cases that depend on it. Prosecutors in general have broad discretion over whom to charge, and not moving a case forward is the exercise of such discretion.[9]

This opinion piece is therefore written for people on both sides. Prosecutors and investigators can use it to see what evidence matters, and developers and their counsel can use it to see where the legal exposure lies before designing their systems.

---

[6] For the overt act requirement, and for the conspiracy statutes that dispense with it, see the statute and decisions cited at footnotes 12, 15, and 16. For the knowledge and intent elements see footnote 4.

[7] That a Safe transaction can be approved by signatures made away from the blockchain, with no transaction from any owner, and that the owner signatures travel in the call that executes it and so become public, from which the signing blockchain address can be recovered, rest on the Safe contract source cited at footnotes 30, 31, and 38 and on the Safe documentation cited at footnote 32. What the published signatures do and do not show is set out in Section 4.5.

[8] Safe Docs, "Smart Account Concepts," https://docs.safe.global/advanced/smart-account-concepts (accessed October 26, 2026). The page states that the Safe smart account supports "relaying by making the confirmation/verification logic independent of `msg.sender`."

[9] *Wayte v. United States*, 470 U.S. 598, 607 (1985) ("In our criminal justice system, the Government retains 'broad discretion' as to whom to prosecute.").

1. SCOPE OF THIS OPINION PIECE

This opinion piece covers two elements of conspiracy law only, the agreement and the overt act. This opinion piece does not ask whether any specific criminal statute applies to any particular blockchain system, and does not cover jurisdiction. Nor is this opinion piece a full legal treatment, because no case is analyzed in depth. The main contribution is the proposed evasion method in Section 4 rather than the legal analysis.

The argument is narrow.

When a group of people operate a system that is governed by voting, and the system is facilitating criminal activity, anyone who sends a permissionless transaction that advances the operation of the system ought to count as a member of the criminal conspiracy. That conclusion holds where the person joined the agreement knowingly and intended to further its criminal object. If the system is facilitating no criminal activity, the case this opinion piece considers does not arise. That qualification states the scope of this opinion piece rather than a proposition about when a conspiracy exists, because a conspiracy need not achieve its criminal object for it to result in legal consequences.[10]

This opinion piece therefore focuses on deployment transactions which directly cause a change in how the system deployed to the blockchain operates and does not consider whether merely voting on a proposal falls within the same scope.

A deployment transaction by itself advances the system forward. Whereas a single vote is one of many contributions to a group decision and may not by itself have decided the outcome for the system deployed to the blockchain. We do not explore that difference, and the point here is only to make the scope clear.

This opinion piece uses United States federal law throughout. Where the opinion piece cites a decision of the Supreme Court, the citation is to the United States Reports, with the page of the quoted passage. *Carlson v. United States* is cited to the Federal Reporter, because that decision is a decision of a court of appeals. *United States v. Storm* is cited by docket number and filing date, because the filings in that prosecution are not published in a reporter. Where the opinion piece cites software, the citation is to the published documentation or to the published source code at a named release tag, with the date on which the page was read. Two documents of the Safe contract repository cited in this opinion piece sit on the development branch rather than at a release tag, and each is pinned to the commit at which it was read. This opinion piece also cites configuration and implementation files of the Discourse repository on that project's main branch, which moves, and each of those is cited with the date on which it was read rather than pinned.

2. CONSPIRACY BASICS

Now that the logistics of this opinion piece are out of the way, let's start with a simple example of a criminal conspiracy. Three friends agree to rob a bank, and thus far nothing has happened except talk. The next morning one of the three buys a mask, and under the general federal conspiracy statute that purchase is an overt act. From that moment all three friends can be charged with conspiracy, even though no bank has been robbed and only one friend purchased anything pursuant to the planned bank robbery.[11]

The general federal conspiracy statute is 18 U.S.C. § 371, which reaches both an agreement to commit a federal offense and an agreement to defraud the United States. For the offense branch the statute

[10] That a conspiracy need not achieve its object is established by the authorities cited at footnotes 4 and 11. The text of 18 U.S.C. § 371 is at footnote 3.

[11] *Braverman v. United States*, 317 U.S. 49, 53 (1942) ("The overt act, without proof of which a charge of conspiracy cannot be submitted to the jury, may be that of only a single one of the conspirators and need not be itself a crime.").

info@chainargos.com

imposes two requirements. First, two or more people must agree to commit a federal offense. Second, at least one of those people must "do any act to effect the object of the conspiracy."[12]

Those two requirements are not the whole of the offense, and the rest of this opinion piece assumes the others rather than restating them. The government must also prove that the defendant joined the agreement knowingly and did so intending to further its criminal object. A person whose act happens to help a conspiracy, while knowing nothing of the agreement and intending nothing by it, is not a conspirator. Every question this opinion piece raises about the sender of a deployment transaction sits on top of that requirement rather than in place of it.[13]

The overt act does not need to be a crime by itself, and buying a mask is perfectly legal. The overt act need only demonstrate that the plan has moved from mere discussion into concrete action to forward an illegal activity. The US Supreme Court has said that the function of the overt act "is simply to manifest 'that the conspiracy is at work,'" and that the plan is "neither a project still resting solely in the minds of the conspirators nor a fully completed operation no longer in existence."[14]

Not every conspiracy statute requires an overt act, and the money laundering conspiracy statute, 18 U.S.C. § 1956(h), does not require one.[15] Neither does the drug conspiracy statute, 21 U.S.C. § 846.[16] Under those statutes the agreement alone is the crime, although the government must still prove that the defendant joined the agreement which is usually with evidence of what the defendant did.[17]

One point about the overt act should be stated plainly, because it bears on Section 6, and that is that the requirement for an overt act is easily met. The act need not be a crime, and it need not be committed by the defendant who is charged. Buying the mask in our earlier example therefore satisfies the requirement for all three friends at once.[18]

The distinction matters for this opinion piece.

Where the charge is brought under § 371, the question is whether sending the deployment transaction is an overt act, and where the charge is brought under § 1956(h) or a similar statute, the question is whether sending it is evidence that the sender joined the agreement. Either way, the deployment transaction is the only transaction that carries out a Safe transaction, a point Section 6 takes up again.[19]

One more rule matters.

The government does not need to know who all the conspirators are. The Supreme Court has observed that one person can be convicted of conspiring with people whose names are unknown.[20] The government does, however, need to prove that at least two people were involved, because a person cannot conspire alone[21] and Section 5 builds on this rule.

---

[12] 18 U.S.C. § 371, https://www.govinfo.gov/link/uscode/18/371.

[13] For the knowledge and intent elements see footnote 4. The text of 18 U.S.C. § 371 is at footnote 3.

[14] *Yates v. United States*, 354 U.S. 298, 334 (1957). The Court added in the same passage that the overt act need not "even be criminal in character."

[15] *Whitfield v. United States*, 543 U.S. 209, 219 (2005) ("[W]e hold that conviction for conspiracy to commit money laundering, in violation of 18 U. S. C. § 1956(h), does not require proof of an overt act in furtherance of the conspiracy."). The text of § 1956(h) is at https://www.govinfo.gov/link/uscode/18/1956.

[16] *United States v. Shabani*, 513 U.S. 10, 15 (1994) ("In order to establish a violation of 21 U. S. C. § 846, the Government need not prove the commission of any overt acts in furtherance of the conspiracy.").

[17] That the government must still prove that the defendant joined the agreement knowingly and intended to further its criminal object is established by the authorities cited at footnote 4. The holdings that those two statutes require no overt act are at footnotes 15 and 16.

[18] *Braverman v. United States*, 317 U.S. 49, 53 (1942), and *Yates v. United States*, 354 U.S. 298, 334 (1957), both quoted above in this section.

[19] That the submitter's call is the transaction that carries out a Safe transaction rests on the Safe contract source cited at footnotes 30 and 31. For the distinctions between the conspiracy statutes drawn in this paragraph see the statute cited at footnote 12 and the decisions cited at footnotes 15 and 16.

[20] *Rogers v. United States*, 340 U.S. 367, 375 (1951) ("Of course, at least two persons are required to constitute a conspiracy, but the identity of the other members of the conspiracy is not needed, inasmuch as one person can be convicted of conspiring with persons whose names are unknown."). *Rogers* is a self-incrimination and contempt case, and the quoted passage is reasoning rather than a holding on conspiracy law.

[21] *Morrison v. California*, 291 U.S. 82, 92 (1934) ("It is impossible in the nature of things for a man to conspire with himself.").

3. HOW A SAFE MULTISIG WORKS

A group controlling a Safe can separate approval from execution. Owners of a Safe approve an action by signing data away from the blockchain. A submitter then sends those signatures to the Safe in a blockchain transaction. Signing does not itself create a transaction from an owner, but the signatures become public when submitted. The blockchain therefore records both the submitter's transaction and the owner keys that signed. It does not by itself establish who held each key. This section explains that distinction and the checks the Safe performs pursuant to such a transaction.

A multisig, short for "multiple signatures," is a smart contract that requires a set number of approvals before a transaction is carried out. Think of a bank vault with five keyholes that opens when any three keys are turned. The vault does not care which three. It only counts the number of keys.[22]

The multisig product this opinion piece considers is the Safe, which was previously called the Gnosis Safe until 2022. A community vote in April 2022 approved splitting the project off from Gnosis Ltd., and the rebranding to Safe was announced on July 12, 2022. The Safe project reports more than 63 million accounts deployed and more than 1.1 billion transactions, as figures for the second quarter of 2026.[23] The Safe is a set of smart contracts that developers use to build tools in which a set number of signatures made with private keys is required before crypto-assets move or before a system changes. Those figures are the project's own, and this opinion piece makes no claim about which multisig product is used most. They are given because the behavior described below is the behavior of a widely deployed product rather than of a special case.

A few more words need defining here. A private key is a large secret number, and whoever knows it can produce digital signatures. A digital signature is a short piece of data proving that the holder of the private key approved a specific message. Anyone can check such a signature, and no one can produce one for a new message without the private key. A signature therefore shows that the key was used, which is not the same as showing that a particular person decided to use it. A private key produces a public key, and a blockchain address is a shorter identifier computed from that public key. The blockchain address is public whereas the private key is not.[24]

A signet ring illustrates the relationship. The ring is the private key, and the pattern it presses into wax is the public blockchain address. Anyone can compare a wax seal against the known pattern, but only the ring can make the seal.

In the Safe, the parties that can approve a Safe transaction are called owners, and each owner is a blockchain address. An owner can be a person holding a private key, or another smart contract. The number of approvals required before a Safe transaction is carried out is called the threshold, which the Safe contract describes as the "Number of required confirmations for a Safe transaction." A

---

[22] For the Safe as a smart contract that requires a set number of approvals see the Safe documentation cited at footnote 25 and the owner manager source cited at footnote 27.

[23] SiliconANGLE, "Gnosis Safe rebrands as Safe and raises $100M for digital asset management," July 12, 2022, https://siliconangle.com/2022/07/12/gnosis-safe-rebrands-safe-raises-100m-digital-asset-management/ ("Digital asset management platform Gnosis Safe announced today that it has raised $100 million in a token sale and rebranded as Safe as the community voted to split from its parent company Gnosis Ltd."). The governance proposal is GIP-29, at https://forum.gnosis.io/t/gip-29-spin-off-safedao-and-launch-safe-token/3476 (accessed November 2, 2026). Post 58 of that thread, dated April 10, 2022, records that voting was active and links the recorded vote at https://snapshot.org/#/gnosis.eth/proposal/0x5187694133abfb3d140acce3b9ac2ecf033c657c816a18dc25d2e0c1a2a29ec7. Post 61, dated April 16, 2022, records that the proposal passed. The voting weights were not obtained for this opinion piece. For the adoption described in the text, the Safe Foundation publishes figures headed "Figures as of Q2 2026" giving "63M+" accounts deployed, "$27B+" assets secured, and "1.1B+" transactions, at https://safefoundation.org/smart-contracts (accessed November 1, 2026). Those are the project's own figures rather than an independent measurement, the page gives a quarter rather than a date, and it does not state how many of the accounts sit on the Ethereum blockchain. No third party measurement of the share of multisig use held by any product was obtained for this opinion piece, so this opinion piece makes no claim about which product is used most.

[24] For digital signatures as the Safe uses them, and for the recovery of a blockchain address from a signature, see the Safe documentation cited at footnotes 32 and 67. For the derivation of a public key from a private key, and of a blockchain address from a public key, see Ethereum Foundation, Ethereum Developer Documentation, "Ethereum accounts," https://ethereum.org/developers/docs/accounts/ (accessed November 10, 2026). The page carries a page update date of August 6, 2026. The page describes an externally owned account as "controlled by anyone with the private keys," states that "[t]he public key is generated from the private key using the Elliptic Curve Digital Signature Algorithm," and states that a blockchain address is obtained "by taking the last 20 bytes of the Keccak-256 hash of the public key and adding `0x` to the beginning." That page describes the derivation rather than the mathematics of the signature scheme, and no separate source for the scheme is cited in this opinion piece.

confirmation is usually a signature, although Section 4.1 describes one route by which an owner counts toward the signature threshold without producing a fresh signature.[25] A Safe might require three approvals out of five owners, or five out of seven.

The owners can collectively add owners, remove owners, and change the threshold of approvals required, although on the ordinary route each of those changes is itself a Safe transaction that must meet the current threshold of approvals. A change to who controls a Safe is therefore made by the same route as anything else a Safe does, and that is the route the evasion method in Section 4 uses. [26]

A function is a named operation that a smart contract can carry out. A call is an instruction to a smart contract to carry out one of its functions. In the Safe contract code, the functions are "`addOwnerWithThreshold`", "`removeOwner`", "`swapOwner`", and "`changeThreshold`".[27] Each function carries a requirement named "`authorized`", which makes the transaction fail unless the party calling the function is the Safe contract itself.[28] In plain words, only the Safe can change the Safe. Where the change comes through an approved Safe transaction, the Safe acts only when enough owners have given their approval. The Appendix describes a route that reaches the same functions with no approval at the time of acting.[29]

Figure 1 separates owner approval from submission, then shows the main checks the Safe applies before attempting the approved action. The rest of this section explains those checks.

[25] Safe Docs, "Smart Account Concepts," https://docs.safe.global/advanced/smart-account-concepts (accessed October 26, 2026). The page states that "[e]ach Safe account maintains its own list of owners in the storage in the form of Ethereum addresses" and that the Safe "maintains a threshold, the minimum number of owners required to confirm a transaction before it gets executed." The quoted header definition of the threshold comes from `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, which is a different source from the documentation page.

[26] The owner management functions, and the requirement that each change meet the current threshold, are in the owner manager source cited at footnote 27 and the self-call requirement cited at footnote 28. The threshold check itself sits in `Safe.sol`, cited at footnotes 30 and 31.

[27] `OwnerManager.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/base/OwnerManager.sol (accessed October 26, 2026). Each of the four functions is declared "`public override authorized`".

[28] `SelfAuthorized.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/common/SelfAuthorized.sol (accessed October 26, 2026). The requirement calls "`requireSelfCall()`", which reads "`if (msg.sender != address(this)) revertWithError("GS031");`".

[29] That only the Safe can call the owner management functions rests on the self-call requirement cited at footnote 28. The route that reaches those functions with no contemporaneous threshold check is described in the Appendix and supported at footnotes 120 and 122.

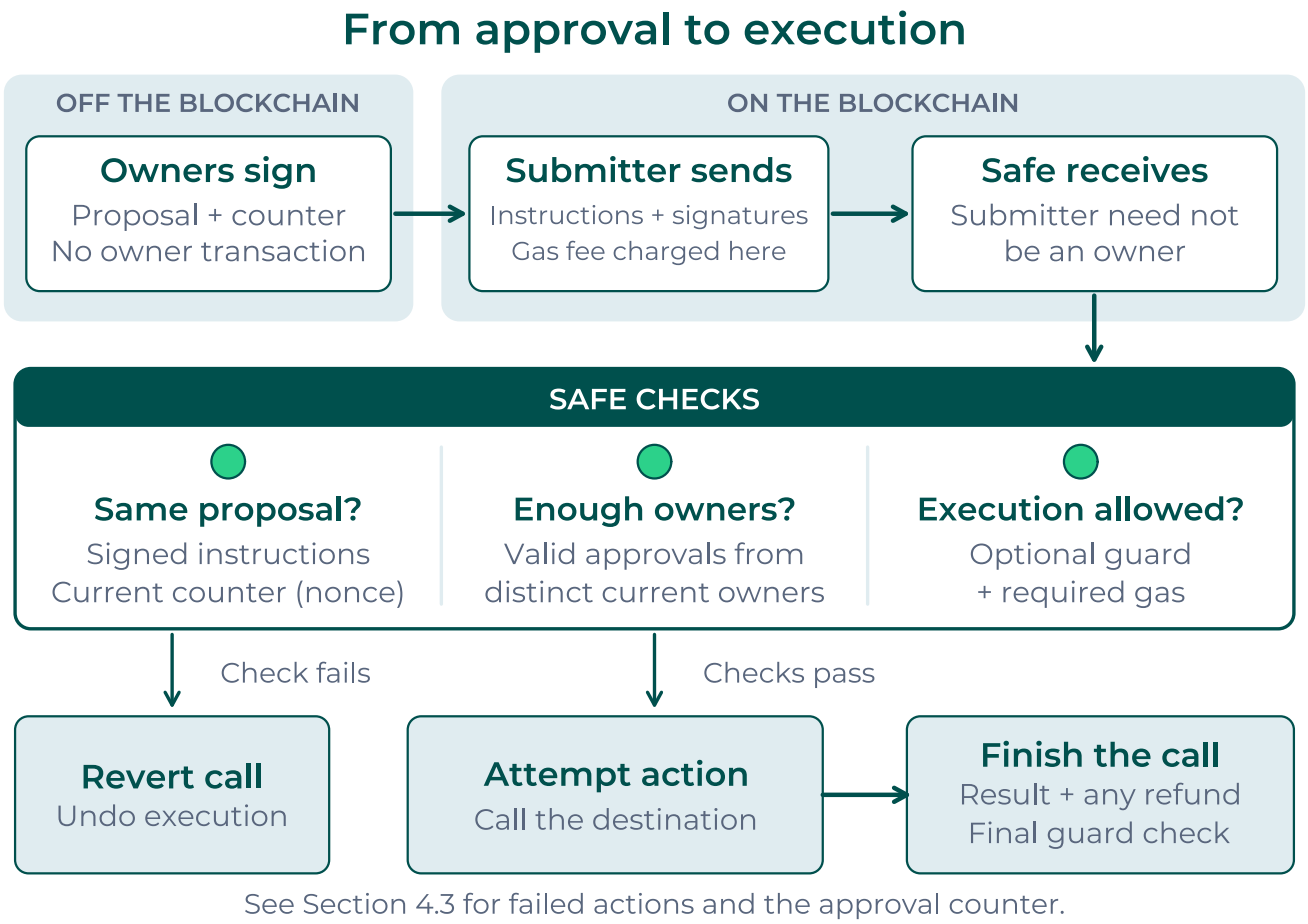


Figure 1. The ordinary approval route. Owners sign, a submitter sends the transaction, and the Safe checks approval and execution conditions. The diagram summarizes the process. Section 4.3 explains failure and nonce consumption.[30]

The submitter calls the Safe's "`execTransaction`" function with the approved instructions and the owner signatures. The instructions specify what the Safe should do, including the destination blockchain address for the transaction, the crypto-asset amount, any data to send, and the type of operation. They also specify gas settings and any gas refund from the Safe's holdings to the designated recipient. On Ethereum, the blockchain's own crypto-asset is ether. The gas settings are "`safeTxGas`", "`baseGas`", and "`gasPrice`". Their effects depend on the combination chosen, as Section 4.3 explains. The exact fields are

[30] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed November 4, 2026). The figure is drawn from the body of "`execTransaction`" at lines 126 to 205, and from "`checkNSignatures`" at lines 287 to 342. The figure summarizes approval, submission, the principal Safe checks, and execution. It does not depict every internal step or the detailed failure branches. The nonce is increased before the signature check, as described below. Any required check can revert the whole call, including the final guard check. Destination failure may instead be retained under the conditions at footnote 59. The source was checked again on November 14, 2026. The stored nonce is increased at line 145, as the argument to "`getTransactionHash`", and the signature check follows at line 147. The owner list and ordering check sits inside the loop that reads the entries, at the end of each iteration, at lines 338 and 339, and refuses the entry with error `GS026`. The gas fee for the Ethereum transaction that carries the call is charged by the Ethereum blockchain to the blockchain address that sends it, and not by the Safe. For gas and the fee calculation see the Ethereum developer documentation cited at footnotes 31 and 32. That documentation does not state in general terms which party bears the fee. That documentation shows the charge falling on the sending blockchain address in a worked example, and it states that "[t]he transaction sender is refunded the difference between the max fee and the sum of the base fee and tip," which carries the same consequence. The separate refund the Safe can make is described at footnote 54. The values of the operation field are declared in `Enum.sol`, quoted at footnote 123.

listed in the footnote.[31] The submitter supplies the signatures together, with at least enough valid owner approvals to meet the threshold. Signatures from the remaining owners are not needed.[32]

The Safe calculates a transaction hash, a compact value derived from the proposed action under a public standard called EIP-712.[33] Changing a signed instruction changes the transaction hash. The signed instructions cover the destination blockchain address, amount of crypto-asset to be sent, any data, operation, gas and refund settings, and a counter called the nonce. The nonce distinguishes this specific approval from an otherwise identical action proposed later.

The signatures themselves are not included in the signed transaction hash. The transaction hash also does not include the fields of the Ethereum transaction that carries the call, and it does not require identification of the blockchain address sending that call.[34] An owner's signature therefore approves a transaction and does not authorize any particular party to send it. Any party holding enough valid signatures may attempt to submit the Safe transaction, subject to the other checks described below. Nothing the owners signed records which party submitted it.

The nonce increases by one each time an ordinary Safe transaction runs and the blockchain keeps the call that carried it. The Safe contract header explains that "[e]ach transaction should have a different nonce to prevent replay attacks."[35]

The Safe contract then checks entries in the sequence against the hash, taking them in order and stopping once it has checked as many entries as the threshold requires. Entries supplied beyond that number are not examined. The submitter assembles the sequence, so where more owners have signed

---

[31] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. The declaration reads "`function execTransaction(address to, uint256 value, bytes calldata data, Enum.Operation operation, uint256 safeTxGas, uint256 baseGas, uint256 gasPrice, address gasToken, address payable refundReceiver, bytes memory signatures) external payable override returns (bool success)`". See also Safe Docs, Smart Account Reference, "`execTransaction`," https://docs.safe.global/reference-smart-account/transactions/execTransaction (accessed October 26, 2026). That page describes "`safeTxGas`" as "Gas that should be used for the Safe transaction," "`baseGas`" as "Gas costs that are independent of the transaction execution," and "`gasPrice`" as "Gas price that should be used for the payment calculation," and it was read again on November 13, 2026. The page gives no Safe contract version. The source at release tag v1.5.0 was retrieved and checked on November 14, 2026. The refund call at line 197 and the comments at lines 212 and 213 confirm the roles of "`baseGas`" and "`gasPrice`". That "`safeTxGas`" is the allowance applied to the call the Safe carries out rests on the source itself, quoted at footnote 59. Code quotations in this footnote are reformatted onto one line from the source. That ether is the crypto-asset of the Ethereum blockchain is stated by Ethereum Foundation, Ethereum Developer Documentation, "Gas and fees," https://ethereum.org/developers/docs/gas/ (accessed November 10, 2026), which carries a page update date of October 23, 2026, and which states that "[g]as fees have to be paid in Ethereum's native currency, ether (ETH)." The apostrophe in that quotation is reproduced as a typographic mark, and the source renders it as a plain mark.

[32] Safe Docs, "Signatures," https://docs.safe.global/smart-account-signatures (accessed October 26, 2026). The page states that "[a]ll signatures are combined into a single `bytes` and transmitted to the contract when a transaction should be executed" and that each signature has a constant part of 65 bytes. The 65 bytes describe the constant part only. A smart contract signature made under EIP-1271, which is the published standard by which a smart contract rather than a private key holder can present a signature, appends further data beyond its constant part, so a reader assembling signatures for a Safe that has a smart contract owner cannot assume a flat 65 bytes per owner. For gas as the unit in which the blockchain meters work, and for the fee calculation, see the Ethereum developer documentation cited at footnote 31, which states that "[g]as refers to the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network" and that "[t]he gas fee is the amount of gas used to do some operation, multiplied by the cost per unit gas." The same page gives the current form of the calculation as "units of gas used * (base fee + priority fee)", in which the base fee is set by the protocol and the priority fee is set by the sender. The gas price described in the text is the effective price charged for each unit, and not the maximum a sender is willing to bear.

[33] Remco Bloemen, Leonid Logvinov, and Jacob Evans, "EIP-712: Typed structured data hashing and signing" (Final), https://eips.ethereum.org/EIPS/eip-712. The abstract describes the standard as "a standard for hashing and signing of typed structured data as opposed to just bytestrings." The standard is cited in its final form at the location given above, read on November 1, 2026. That this Safe implementation builds its transaction hash under that standard is established by the Safe contract source cited at footnotes 30 and 34.

[34] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. The Safe contract defines "`SAFE_TX_TYPEHASH`" as the precomputed hash under "`keccak256`" of "`SafeTx(address to,uint256 value,bytes data,uint8 operation,uint256 safeTxGas,uint256 baseGas,uint256 gasPrice,address gasToken,address refundReceiver,uint256 nonce)`".

[35] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. The code reads "`txHash = getTransactionHash(..., nonce++);`" with the comment "We use the post-increment here, so the current nonce value is used and incremented afterwards." The code quotation is reformatted onto one line, and the ellipsis stands for arguments omitted here rather than appearing in the source.

than the threshold requires, the submitter decides which owner signatures appear in the call the blockchain keeps. For each entry in the sequence, the Safe contract establishes which owner the entry belongs to. For an ordinary signature the Safe contract recovers the blockchain address from the signature itself. [36]

Two other kinds of entry exist, which are the pre-approved hash described in Section 4.1 and a signature made by a smart contract. For those the Safe contract reads the blockchain address out of the entry rather than recovering it. The Safe contract then confirms that the blockchain address is on the owner list, and confirms that the entries are sorted in ascending order of owner blockchain address so that no owner is counted twice.[37] Reaching the threshold satisfies the approval check. Execution remains subject to the other checks described below.

Note what the Safe contract does not require.

The Safe contract does read who sent the "`execTransaction`" call, a party the software code refers to as "`msg.sender`". The Safe contract passes "`msg.sender`" into the signature check and to any guard (see below). What the Safe contract does not do is require that party to be an owner.[38] The Safe documentation says the same thing in different words. The Safe supports "relaying by making the confirmation/verification logic independent of '`msg.sender`'."[39]

Two qualifications belong with that statement. First, a Safe may have a guard installed. A guard is an optional smart contract that the Safe calls before carrying out a transaction, and the Safe passes the caller to the guard. A guard can therefore refuse a transaction based on who the caller is. A Safe with such a guard does restrict who may send the call.[40] Second, one kind of signature entry rests on something other than a fresh signature. An entry of that kind counts where the owner named in the entry has pre-approved the transaction hash in the way described in Section 4.1, and in that case any party may submit it. An entry of that kind also counts where the owner named in the entry is the party sending the call. Only where neither holds does the Safe contract refuse the entry, with error `GS025`.[41] The proposed evasion method in Section 4.2 does not use that kind of entry, and as such is unaffected by the second qualification.

Back to the vault. The vault checks the keys, not who carried them to the door.

Return to the signet ring. A sealed letter can be carried by any courier, who can neither change what is inside without breaking the seal nor make a new seal. The courier need not be one of the people who sealed the letter. The clerk who receives the letter checks the seal against the known pattern and acts

---

[36] The order in which the Safe reads the entries, the point at which it stops, and the recovery of a blockchain address from an ordinary signature are in the Safe contract source cited at footnotes 30 and 37.

[37] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. The check in "`checkNSignatures`" reads "`if (currentOwner <= lastOwner || owners[currentOwner] == address(0) || currentOwner == SENTINEL_OWNERS) revertWithError("GS026");`". See also Safe Docs, "Smart Account Concepts," https://docs.safe.global/advanced/smart-account-concepts (accessed October 26, 2026). ("`execTransaction` expects that the signatures are sorted by owner address. This is required to easily validate no confirmation duplicates exist."). The code quotation is reformatted onto one line. The documentation sentence ends with a closing parenthesis in the source, which is omitted here because the parenthesis belongs to the surrounding text rather than to the quoted sentence.

[38] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. Within "`execTransaction`", "`msg.sender`" appears only in "`checkSignatures(msg.sender, txHash, signatures);`" and as an argument to the optional transaction guard. The only place the executor matters is the approved-hash branch of "`checkNSignatures`", which reads "`if (executor != currentOwner && approvedHashes[currentOwner][dataHash] == 0) revertWithError("GS025");`".

[39] Safe Docs, "Smart Account Concepts," https://docs.safe.global/advanced/smart-account-concepts (accessed October 26, 2026).

[40] `Safe.sol` at release tag v1.5.0 passes "`msg.sender`" as the final argument to "`ITransactionGuard(guard).checkTransaction`". A Safe with no guard installed makes no such call.

[41] `Safe.sol` at release tag v1.5.0, "`checkNSignatures`". The approved hash branch reads "`if (executor != currentOwner && approvedHashes[currentOwner][dataHash] == 0) revertWithError("GS025");`".

on the letter, without needing to know the courier's name. Where the office has standing instructions to turn particular couriers away, that instruction is the guard described above.

3.1 Two further ways a Safe can act

A Safe can also act through features that do not require fresh owner approvals or that can change the rules governing the Safe itself. While these alternatives still require someone, for instance the submitter, to send a blockchain transaction, the owner list of the Safe alone does not establish who approved the transaction. The Appendix explains these alternatives and their limits, while the rest of this opinion piece considers the ordinary approval and submission process.

Four facts from this section carry the rest of the argument. First, the owners of a Safe are blockchain addresses rather than named people. Second, an owner on the ordinary route for Safe operation approves by producing a digital signature, which is data rather than a transaction, so approving does not produce a blockchain transaction record. Third, the owner signatures are supplied inside the call that makes the Safe act. Fourth, the Safe does not require the party that sends that call to be an owner, and on every route described in Section 3 and the Appendix a party must send a transaction. The Safe contract source and documentation cited in Section 3 and the Appendix support these four points.

4. SIGNING WITHOUT AN OWNER TRANSACTION

In this section, we will explore a proposed evasion method step by step. No step departs from the way a Safe is built to work, requires any modification of the existing Safe software, or any other form of tampering, and Section 3 has already described every feature the proposed evasion method uses.

The result of this proposed evasion method is such that a group can operate a smart contract system, decide changes to that system by vote, and carry each change out, while no owner blockchain address ever sends a transaction. The submitter sends the transaction that carries out each approved action and need not be part of the group that approved that transaction. Setup and transfers supplying gas fees can produce additional blockchain transactions. Sections 4.1 and 4.2 set out two approval routes. The first requires a blockchain transaction from each approving owner. The second requires no owner transaction, but the owner blockchain addresses and submitted signatures remain public.

4.1 Recording approval on the blockchain

One way a multisig can operate is to gather enough owners to meet the threshold where each of the owners sends a transaction to the blockchain saying "I approve," and someone then sends a final transaction carrying out the Safe transaction. Each approving owner therefore leaves a visible entry in the blockchain ledger.[42]

The visible approach is one way the Safe can work, and the Safe documentation calls it "pre-validated signatures." An owner calls a function named "`approveHash`" in a separate transaction, and the Safe records that the owner has approved that specific transaction hash. When "`execTransaction`" later runs, a signature of a special type, marked with the value 1, tells the Safe contract to look up the recorded approval rather than check a cryptographic signature.[43] The function "`approveHash`" only accepts calls

---

[42] That an approval given in advance is itself a transaction sent by the owner is established by the approval function cited at footnote 44 and the Safe documentation cited at footnote 43.

[43] Safe Docs, "Signatures," https://docs.safe.global/smart-account-signatures (accessed October 26, 2026). The page states that "[t]he Safe keeps track of all hashes that have been pre-validated" and that "[t]o add an entry to this mapping use `approveHash`." The signature type for this case is "`signature type == 1`". The same page describes the storage as "a mapping address to mapping of bytes32 to boolean." That description is inaccurate. The Safe contract declares "`mapping(address => mapping(bytes32 => uint256)) public override approvedHashes;`", and "`approveHash`" writes the value 1. `Safe.sol` at release tag v1.5.0. Where the documentation and the Safe contract source differ, the Safe contract source governs.

from an owner.[44] An owner who is also the sender of the "`execTransaction`" call does not need to call "`approveHash`" first, because the sender's own call counts as approval.[45]

An owner who sends a separate "`approveHash`" transaction leaves a public record of consent to that transaction hash. The blockchain charges the gas fee to the owner blockchain address that sends the transaction, so that blockchain address must hold the crypto-asset needed for the fee.[46] Figure 2 compares this route with the signatures described next.

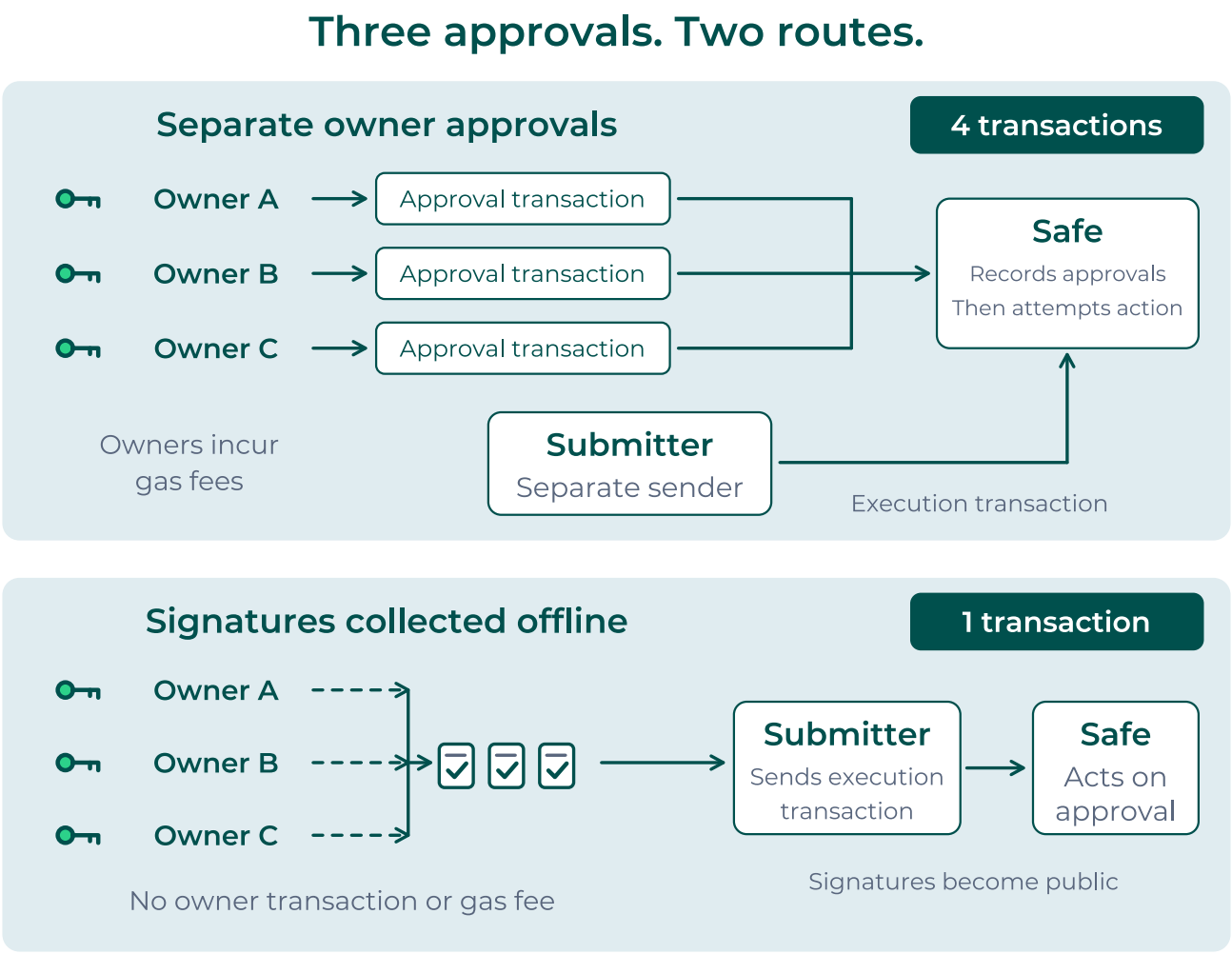


Figure 2. Three approving owner blockchain addresses and a separate submitter. Separate approval transactions require four transactions in total. Collecting signatures offline requires one execution transaction. Setup and gas-fee transfers are excluded.[42, 54]

---

[44] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. The function reads "`function approveHash(bytes32 hashToApprove) external override { if (owners[msg.sender] == address(0)) revertWithError("GS030"); approvedHashes[msg.sender][hashToApprove] = 1; emit ApproveHash(hashToApprove, msg.sender); }`". Code quotations in this footnote are reformatted onto one line from the source.

[45] Safe Docs, "Signatures," https://docs.safe.global/smart-account-signatures (accessed October 26, 2026). The page states that "if the validator is the sender of transaction that executed the Safe transaction it's not required to use `approveHash` to add an entry to the mapping."

[46] That the approval is recorded against the owner's own blockchain address, and that the owner must send a transaction to give it, are in the approval function cited at footnote 44. That the sender of a transaction bears its gas fee follows from the Ethereum developer documentation cited at footnotes 31 and 32, subject to the limit recorded at footnote 30.

4.2 Approving away from the blockchain

The second route produces signatures without blockchain transactions from the owners. The signatures become public when the submitter uses them. The six steps below separate the owners' approval from the submitter's act of execution.

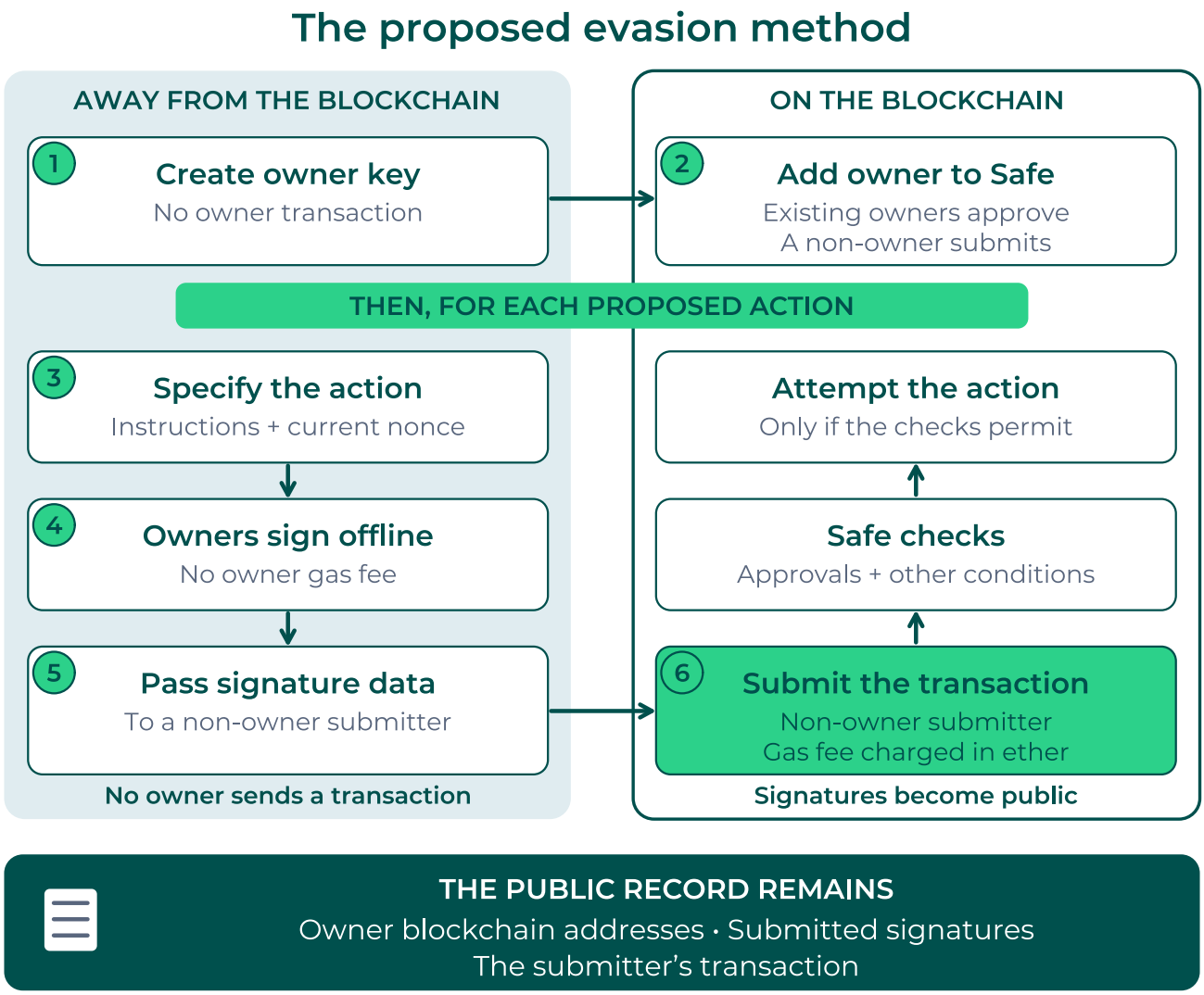


Figure 2A. The six steps in the proposed method. A non-owner submits both the owner addition and the later action. Owner blockchain addresses and submitted signatures remain public. The gas fee and any Safe refund are explained after step six.[48, 54, 69]

Step One: Create an Owner Key

A private key is a random number, and the blockchain address is computed from it by arithmetic alone, so no transaction is needed. The blockchain address exists the moment the arithmetic is done, and a

info@chainargos.com

person following these steps should confirm that it has never appeared on the blockchain and holds no crypto-assets.[47]

Step Two: Add the Newly-Created Blockchain Address to the Safe

The existing owners approve a Safe transaction calling "`addOwnerWithThreshold`" with the new blockchain address, and a submitter sends it to the Safe.[48] The new owner does not have to send a transaction or hold crypto-assets to be added. An unused blockchain address in the owner list is therefore not, by itself, evidence that the Safe is misconfigured.

Step Three: Specify the Action to Approve

The proposed action includes the instructions described in Section 3 and the current nonce. Those values determine the transaction hash under EIP-712. Anyone holding the same values can calculate the same transaction hash, so owners can check exactly which action they are being asked to approve.[49]

Step Four: Sign the Approval

Each approving owner uses the private key to sign the transaction hash. The result is a short piece of data.[50] Signing does not in and of itself send any transaction to the blockchain or incur a gas fee.

Step Five: Pass the Signatures to a Submitter

The owners pass the signature data to a submitter, who need not be an owner. The Safe Transaction Service provides one way to collect signatures away from the blockchain.[51] The Safe's software development kit combines the signatures and sorts them by their signing blockchain addresses.[52] These are features of the Safe's own tools.[53]

---

[47] That a public key is derived from a private key, and a blockchain address from a public key, is established by the Ethereum developer documentation cited at footnote 24, which also states that "[c]reating an account costs nothing." That page does not state whether bringing such an account into existence requires a transaction, so the statement in the text that no transaction is needed rests on the arithmetic character of the derivation rather than on an express statement in the source. Arithmetic alone does not establish that a particular blockchain address has never received an unsolicited transfer, which is why the text directs a person following these steps to look the blockchain address up first.

[48] `OwnerManager.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/base/OwnerManager.sol (accessed October 26, 2026).

[49] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. "`SAFE_TX_TYPEHASH`", quoted in Section 3.

[50] Safe Docs, "Signatures," https://docs.safe.global/smart-account-signatures (accessed October 26, 2026). For an ECDSA signature the constant part is "{32-bytes r}{32-bytes s}{1-byte v}", and the signature type byte encodes "`v`" in the range "31 > signature type > 26".

[51] Safe Docs, "Safe Transaction Service," https://docs.safe.global/core-api/transaction-service-overview (accessed October 26, 2026). The page states that the service "provides endpoints to send transactions, allow off-chain collecting of signatures, or informing the owners about a pending transaction to be sent to the blockchain."

[52] Safe Docs, "Transaction signatures" (Protocol Kit guide), https://docs.safe.global/sdk/protocol-kit/guides/signatures/transactions (accessed October 26, 2026). The guide states that "we need to collect the signatures from the signers who will sign it" and that the Safe Transaction Service is "a centralized and open-source service that anyone can deploy and run" used "To store the transactions and signatures off-chain."

[53] That the Safe's own tools follow the pattern described is established by the Safe documentation cited at footnotes 51 and 52.

Step Six: Submit the Approved Action

A submitter who is not an owner of the Safe sends the approved instructions and the sorted signatures to "`execTransaction`". The blockchain charges the gas fee to that submitter's blockchain address.[54] The Safe checks the signatures and the other conditions described in Section 3. The action executes only if those conditions permit it. To preserve this example's premise that no owner receives crypto-assets, any Safe refund must also go to a blockchain address that is not an owner. If the refund recipient is unset, the Safe uses the sender of that blockchain transaction who is the non-owner submitter in this case. A refund is relevant only if the signed gas price is above zero and a refund is actually made.

The result is that the proposed evasion method in this section requires no transaction from any owner blockchain address, and the only party who sends a transaction is the submitter who calls "`execTransaction`".

The approval requirement concerns the owner signatures, not the submitter's membership in the owner group. Execution still depends on the current nonce, owner list, threshold, signature ordering, gas, any guard, and the other checks described above.[55]

The gas fee and the Safe's refund

Gas measures the computational work used by an Ethereum transaction and the gas fee is the charge for that work. The submitter sets a gas limit, which caps how much gas the Ethereum transaction can use. The final gas fee depends on the gas used and the applicable price per unit. Any unused portion of the gas allowance is handled automatically by Ethereum and is not charged. No separate refund request is needed. The Safe's optional "refund", described below, is a separate transfer from the Safe's holdings.

In this example, Ethereum charges the submitter in ether. The submitter must already hold enough ether to send the transaction, even if a Safe refund is expected. The owners incur no gas fee merely by signing a transaction away from the blockchain.[31, 54]

The Safe's "refund" is a separate transfer from the Safe's holdings to the recipient approved by the owners. It is optional. Setting the Safe's signed "`gasPrice`" to zero disables this transfer, but Ethereum still charges its gas fee. The refund can be in ether or a specified token. A token refund does not change Ethereum's requirement that its own fee be in ether.[31, 54]

The refund need not equal the Ethereum fee. The Safe calculates it using the gas measured during the action, an allowance for other costs, and the approved refund price. The Safe must hold the crypto-asset it transfers. Its holdings are separate from the holdings at each owner's blockchain address.[54]

A failed action can still incur an Ethereum gas fee and, in some circumstances, a Safe refund. If the entire Safe call is undone, the refund is undone too, while the Ethereum gas fee remains. Section 4.3 explains when failure undoes the Safe call.[31, 54, 59]

---

[54] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. "`execTransaction`", quoted in Section 3. The "`refundReceiver`" field permits a gas refund to be directed to any blockchain address, and "[i]f set to `` `0` ``, `` `tx.origin` `` will be used." Safe Docs, "Smart Account Concepts," https://docs.safe.global/advanced/smart-account-concepts (accessed October 26, 2026). The initial gas fee for the Ethereum transaction that carries the call is charged to the blockchain address that sends it, as recorded at footnotes 30 to 32, and the refund described here is a separate transfer the Safe makes afterward. The release-tag source was checked again on November 15, 2026. Lines 177 to 181 measure gas around the action, and lines 195 to 197 call "`handlePayment`" only when "`gasPrice`" is above zero. Lines 225 to 233 select the recipient and calculate the transfer. For ether, the amount is "`gasUsed`" plus "`baseGas`", multiplied by the lower of the signed "`gasPrice`" and "`tx.gasprice`". For a token, the same gas sum is multiplied by the signed "`gasPrice`". These calculations do not guarantee equality with the fee for the whole Ethereum transaction. Lines 229 to 233 revert the Safe call if the required refund transfer fails. For Ethereum's gas fee on an executed but unsuccessful transaction, see the Ethereum documentation cited at footnote 31, checked again on November 15, 2026.

[55] That the submitter needs the signatures, the ordering, and the current nonce, and that no owner need send a transaction, rest on the Safe contract source cited at footnotes 30, 35, and 37.

4.3 What publishing a signature permits

Publishing an ordinary Safe signature reveals approval of particular instructions, not the private key of the Safe's owners. The signed value ties those instructions to the Safe's blockchain address, the blockchain, and the nonce.[56] Once execution consumes that nonce, the signatures cannot execute the same action again on that Safe. This protection assumes that the Safe's code and stored data have not been rewritten through a delegate call of the kind described in the Appendix.[57] A recipient cannot change the signed instructions without invalidating the signature.[58] But anyone holding enough valid signatures can attempt to deliver the approved instructions. The signature does not reserve that right to a named recipient or specify when delivery must occur.

Four limits on that statement matter to anyone relying on it. First, an ordinary Safe signature carries no expiry of its own, because no field of the signed data records a time. Whether the signature can be used at a given moment is a separate question. Usability turns on the nonce still being current, on the signers still being owners, on the signature set still satisfying whatever threshold is then in force, and on nothing else standing in the way, such as a guard or the state of the destination. An unchanged threshold is not required. The signed data does not cover the threshold, and the Safe reads the threshold when the call arrives. Lowering the threshold does not by itself invalidate an existing set that still satisfies the other checks. A party holding a threshold number of signatures can therefore send the call at any time before another Safe transaction consumes the nonce, with no further owner involvement. Publishing a set of signatures is closer to publishing an executable transaction than to publishing a statement of intent.

Second, owners may sign both a transaction and a replacement transaction at the same nonce, and a party holding both sets chooses which one runs.

Third, the signed gas settings affect whether a failed action consumes the approval. With a gas price above zero, the Safe requests the signed gas allowance for the action, subject to Ethereum's gas rules. With a gas price of zero and an allowance above zero, the allowance instead sets a minimum check. The action then receives nearly all the gas remaining in the call. Passing that minimum check does not guarantee enough gas for the action to succeed. If the action fails but the whole Safe call is retained, the nonce is consumed and the owners must sign again. When both the signed gas price and allowance are zero, failure of the action undoes the whole call, leaving the nonce available. The footnote provides the exact code and conditions.[59] Fourth, an ordinary signature can have two different byte representations that recover the same owner. The Safe prevents that owner from being counted twice and uses the nonce to prevent replay. Comparing signature bytes alone can therefore give a misleading result.

---

[56] Remco Bloemen, Leonid Logvinov, and Jacob Evans, "EIP-712: Typed structured data hashing and signing" (Final), https://eips.ethereum.org/EIPS/eip-712. The standard lets the designer of a smart contract choose which fields the signing value covers, and fields that are not used may be left out. Safe version 1.5.0 covers both the smart contract and the identifier of the blockchain, so a signature made for a Safe binds to one smart contract on one blockchain. That coverage is a choice made by the Safe and not a requirement of every signature made under the standard. For the Safe, the structured data is the "`SafeTx`" type quoted in Section 3, which includes the nonce.

[57] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/Safe.sol (accessed October 26, 2026). Line numbers and quotations in this opinion piece are to that release tag, not to the development version of the code, which carries unreleased changes while still reporting "`VERSION`" as 1.5.0. "Each transaction should have a different nonce to prevent replay attacks."

[58] That changing any field of the Safe transaction changes the hash rests on the type definition cited at footnote 34 and the standard cited at footnote 56.

[59] `Safe.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository. The gas check reads "`if (gasleft() < ((safeTxGas << 6) / 63).max(safeTxGas + 2500) + 500) revertWithError("GS010");`". The gas passed to the destination reads "`gasPrice == 0 ? (gasleft() - 2500) : safeTxGas`", so the signed allowance caps the gas only where the gas price is above zero. The comment on that line states that where "the `gasPrice` is 0 we assume that nearly all available gas can be used (it is always more than `safeTxGas`)." Failure is propagated as a revert only where both fields are zero, in a check that reads "`if (!success && safeTxGas == 0 && gasPrice == 0)`". The gas figure the Safe passes is the amount requested when it makes the call. What the destination finally receives is governed by the call costs of the Ethereum blockchain, by the rule in EIP-150 that withholds a sixty-fourth of the gas remaining, and by the stipend added where the call transfers value. The requested figure is therefore not a guarantee of that exact budget. Code quotations in this footnote are reformatted onto one line from the source. See also `safe_tx_gas.md` in the same repository at commit 247b8ce3bdbf204a120e77f5b369d137424ddc58, https://github.com/safe-global/safe-smart-account/blob/247b8ce3bdbf204a120e77f5b369d137424ddc58/docs/safe_tx_gas.md, which describes the no-refund minimum and the risk of setting it too low.

Two further limits are specific to particular versions of the Safe. EIP-712 lets the designer of a smart contract choose which fields the signing value covers. Safe version 1.5.0 chose to cover both the smart contract and the identifier of the blockchain, so a signature made for it ties to one smart contract on one blockchain. Safe version 1.1.1, which was studied for this opinion piece, built that signing value without the identifier of the blockchain. A signature made for such a Safe is therefore executable on another blockchain where a Safe of that version sits at the same blockchain address with the same owners, the same threshold, and the same current nonce.[60] And the Safe project has recorded a change to approved hash signatures, under the heading of allowing such signatures to be bound to a specific transaction hash, which describes behavior present in each of the four releases inspected for this opinion piece. The project's own changelog states that approved hash signatures from the executor "were previously not bound to any particular hash, meaning that submitting the same `execTransaction` call twice could execute it for two different nonces." The cited changelog places the change under its unreleased heading, and it is in any event optional rather than automatic, because a signature that leaves the relevant field empty keeps the existing behavior.[61] The quoted behavior concerns the special approval entry used when an owner is also the caller. That entry can be reused in a later call. It does not mean that an approval stored by "`approveHash`" is unbound, or that a stranger can reuse an ordinary signature for different instructions. Section 4.2 uses ordinary owner signatures and a separate submitter, so it does not depend on this route.[62]

4.4 Passing the signature in public

An owner can publish a signature rather than send it privately to a submitter. The consequence is that anyone who obtains enough valid signatures may attempt the approved action, subject to the limits in Section 4.3. Blockchain governance projects including Uniswap and Arbitrum run public forums on software called Discourse. On a Discourse forum configured for ordinary local signup, with no required identifying custom fields, a new forum account can be created with an email address, a username, a password, and a response to an activation message sent to that email address. The example assumes that the forum requires neither a telephone number nor an identity document. Whether registration is open at all is a setting the forum administrator controls. A Discourse forum can instead be set to

---

[60] That signing value, which the standard calls the domain separator, is in Safe 1.1.1 the hash under "`keccak256`" of "`EIP712Domain(address verifyingContract)`" with the blockchain address of the smart contract. The identifier of the blockchain entered that signing value at version 1.3.0. The domain construction was compared at release tags v1.1.1, v1.3.0, v1.4.1, and v1.5.0. The file is named `GnosisSafe.sol` at v1.3.0 and earlier and `Safe.sol` from v1.4.1 onward. The prototype release v0.0.1 does not use a domain separator at all. `GnosisSafePersonalEdition.sol` and `GnosisSafeTeamEdition.sol` at that release tag each build the transaction hash from a packed payload under a one-byte prefix and a zero byte, with the blockchain address of the smart contract among the packed fields and no identifier of the blockchain, at https://raw.githubusercontent.com/safe-global/safe-smart-account/v0.0.1/contracts/GnosisSafePersonalEdition.sol and https://raw.githubusercontent.com/safe-global/safe-smart-account/v0.0.1/contracts/GnosisSafeTeamEdition.sol (both accessed November 2, 2026). The statement in this footnote about the domain separator is therefore confined to v1.1.1 and later. No survey of every historical tag, and no survey of which implementation is deployed at any particular blockchain address, was carried out.

[61] `CHANGELOG.md` in the safe-global/safe-smart-account repository, under the unreleased heading "Allow binding approved hash signatures to a specific hash," at commit 247b8ce3bdbf204a120e77f5b369d137424ddc58, https://github.com/safe-global/safe-smart-account/blob/247b8ce3bdbf204a120e77f5b369d137424ddc58/CHANGELOG.md (accessed October 30, 2026). The changelog sits on the main branch and moves, so it is pinned to that commit here. Release tags v1.1.1, v1.3.0, v1.4.1, and v1.5.0 were inspected for the approved hash comparison. The statement in the text is limited to those four releases. Footnote 60 records a separate inspection of the prototype release v0.0.1 on the domain separator question.

[62] That the behavior described does not reach the evasion method set out in Section 4.2 rests on the changelog cited at footnote 61 and on the Safe contract source cited at footnote 30.

invitation only, to approval by staff, or to sign in through a third party.[63] At least one email provider offers a free email account and accepts a second email address in place of a telephone number.[64] A VPN, or virtual private network, routes a user's traffic through an encrypted tunnel to a server run by a third party, so that the destination sees the server's internet address rather than the user's own. The operator of that server can still observe and record the connection. An owner can create a single-use email account over a VPN, create a single-use forum account with that email account, post the signature, and never use either the email account or the forum account again. An owner can do this from a cheap telephone on a public wireless internet connection.[65]

The owner does not need to use the same forum account twice. An owner might make one post, under one forum account, arguing that a particular blockchain address should be added as an owner, then later make a second post under a different forum account containing a signature. The posts can establish a link to the same owner key without establishing that one person controlled both forum accounts. Records kept by the forum operator and by the email provider are a separate matter, and the footnote to this section describes what such records have shown in the past.[66]

Checking that a posted signature belongs to the right owner is easy, because anyone can recover the signing blockchain address from the signature and the hash. Recovering the signer from the signature is a property of the signature scheme the Ethereum blockchain uses, and the Safe contract performs the same recovery for an ordinary signature when "`execTransaction`" runs.[67] Recovering the same blockchain address from several signatures links those approvals to a key. It does not prove that one person made all of them, because a key can be shared or transferred. In the signet analogy, the seal shows which ring was used, but not whose hand wore it.

A hotel room key card illustrates what a signature does and does not establish. The lock opens for whoever is holding the card, and the lock records an entry showing which card was used. However, the entry does not show who held the card. A card passed to someone else opens the same door and writes the same entry, so two entries from one card do not establish that one person made both. The private key is the card, and a signature is one use of it.

---

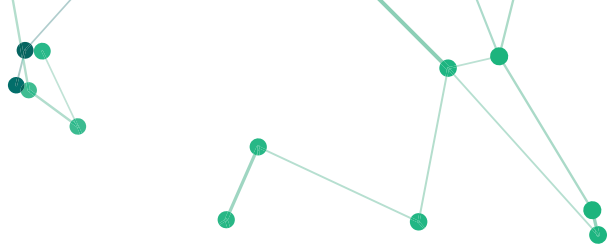
[63] That a Safe signature is safe to publish, and the limits on that statement, are set out in Section 4.3 and in the footnotes to that subsection. Discourse describes itself as a "100% free open source forum software," licensed under version 2 of the GNU General Public License. Discourse, "About," https://discourse.org/about (accessed October 26, 2026). The settings named here that restrict who may join or read a forum are "allow new registrations", "invite only", "must approve users", and "login required". For the first of those four see Discourse Meta, "Disabling Allow New Registrations still shows registration link and modal," https://meta.discourse.org/t/42660, a defect report published on April 16, 2016 in which the setting is named by its administrative label. The current definitions of all four settings are in the Discourse configuration file `config/site_settings.yml` on the main branch, https://raw.githubusercontent.com/discourse/discourse/main/config/site_settings.yml (read November 10, 2026), where the login block declares "`allow_new_registrations`" with a default of true. For the other three see Discourse Meta, "Configuring Discourse for a closed or private community," https://meta.discourse.org/t/27014. Blockchain governance forums that run on Discourse include the Uniswap governance forum at https://gov.uniswap.org/ and the Arbitrum DAO forum at https://forum.arbitrum.foundation/, each of which serves the Discourse generator tag and a "Powered by Discourse" footer. This opinion piece does not state what the registration settings on either of those two forums are. The quoted description of Discourse is the wording on the cited page as read on November 1, 2026, and an earlier wording cannot be reconstructed from the access date alone. The route described in Section 4.4 assumes a forum that permits ordinary local signup and requires no identifying custom fields at signup. Discourse permits custom user fields to be made mandatory at signup, so the setting that allows new registrations does not by itself establish what a given forum asks of a new account. That a field can be made mandatory at signup is established by Discourse, "Creating and configuring custom user fields," https://meta.discourse.org/t/creating-and-configuring-custom-user-fields/113192 (last edited October 8, 2026), which states that where the requirement is set to on signup, "All new account will be required to fill the field," and that an optional field "may be left empty by users." The number disagreement in the first of those two quotations is in the source. This opinion piece does not state how often a forum requires such a field.

[64] Proton Mail, for example, offers free email accounts. Proton, "Create a free email account," https://proton.me/support/create-a-free-email-account-address. Proton lists two human verification methods, a CAPTCHA and a code sent "to your phone number or secondary email address," so a secondary email address can stand in place of a telephone number. Proton, "Human verification," https://proton.me/support/human-verification (accessed October 26, 2026).

[65] For what a VPN does, and for the records a provider can hold, see the record cited at footnote 68.

[66] That the mathematics of the signature is what links two posts, and that nothing else in them need do so, rests on the Safe documentation cited at footnote 67 and on the record of attribution cited at footnote 68.

[67] Safe Docs, "Smart Account Concepts," https://docs.safe.global/advanced/smart-account-concepts (accessed October 26, 2026). ("A Safe Smart Account verifies if each signer is an owner of the Safe account and verifies the signature based on the signature type.") For the implementation of the signature branches, including the recovery of a blockchain address from an ordinary signature, see `Safe.sol` at release tag v1.5.0 cited at footnote 30. The branches that recover a blockchain address from an ordinary signature are at lines 329 to 337, which hold the branch for a signature made under the Ethereum signed message prefix and the default branch, each calling "`ecrecover`". The branch for a hash approved in advance is at lines 325 to 328, and the branch for a smart contract signature ends at line 322.



The proposed evasion method removes transactions sent from owner blockchain addresses. It does not eliminate evidence held by service providers, on devices, or by witnesses and this is why records outside the blockchain can matter to attribution.[68]

4.5 What is left to trace

What is on the blockchain should be stated precisely, because the owner blockchain addresses are not hidden. Each owner blockchain address appears in the data of the transaction that added the owner, in the stored data of the Safe contract, and in the list of owner blockchain addresses the Safe contract gives when asked. Two kinds of public log entry also record owner blockchain addresses. Where an owner is added after the Safe is created, the Safe contract writes an entry naming that owner. Where the owners are installed at the moment the Safe is created, the Safe contract writes a single entry carrying the whole initial owner list. The Safe code comments say that entry exists so that software reading the blockchain can reconstruct how the Safe is configured.[69] Each owner signature also sits permanently in the data of the "`execTransaction`" call that used it, and the owner blockchain address can be recovered from it. Anyone can therefore read the full list of owner blockchain addresses and can read it from the moment the Safe is created.

In the construction described here, no owner blockchain address sends a transaction or receives crypto-assets as part of the process. There is therefore no owner transaction fee or owner-directed transfer trail leading to a crypto-asset exchange. This does not establish that a crypto-asset exchange or another service holds no relevant records about the people involved. Nor can the construction prevent unsolicited transfers to an owner blockchain address. The owner list and submitted signatures remain public evidence even where the owner blockchain addresses have sent no transactions.[70]

One exception should be noted.

---

[68] That statement describes the difficulty of starting from the blockchain or from the forum post. It should be read against the documented instances of users who have been found by other means. Providers have produced records under legal process. HideMyAss produced session records under a United Kingdom court order in the prosecution of Cody Kretsinger in 2011. Its own published description of that disclosure describes records of which customer was connected to which of its server internet addresses at a given time. The same description denies logging the content of a customer's internet traffic, and it describes the monitoring of destination sites as something other providers might do. PureVPN assisted the Federal Bureau of Investigation in the 2017 prosecution of Ryan Lin by showing that one VPN customer account had connected from both his home internet connection and the internet connection at his place of work. Other possible sources of attribution evidence are the billing trail for a VPN subscription or an email account, the reuse of one subscription across identifiable internet connections, the analysis of writing style, the so-called "fingerprinting" of a browser or a device, metadata left in uploaded files, and the records the forum operator keeps. "Fingerprinting" here means the matching of a visitor against a profile built from the characteristics the browser or device reports. A Discourse installation stores the registration email address, the internet address recorded at registration, and a current internet address, and exposes the two internet addresses to staff subject to a permission check. The claim in the text is that none of these starting points is on the blockchain. The HideMyAss description above is taken from the provider's own published statement about the 2011 disclosure, published as "Lulzsec fiasco" at https://blog.hidemyass.com/en/lulzsec-fiasco. That page could not be retrieved from the environment in which this revision was prepared, so it is named here for the reader rather than quoted. A reader consulting it should note that it carries a later update, which is not the policy as it stood in 2011. The PureVPN example rests on the affidavit of Special Agent Jeffrey Williams, sworn October 3, 2017, at paragraph 52, filed with the criminal complaint in the Ryan Lin prosecution, No. 17-MJ-4251-DHH (D. Mass.), published by the Department of Justice at https://www.justice.gov/archives/opa/press-release/file/1001841/dl and read on November 11, 2026. That paragraph records that PureVPN determined that its service had been accessed by the same customer from the internet connection at the home where Mr. Lin was then living and from the software company where he was then employed. It is an investigator's affidavit rather than a finding made after trial, and it does not state what PureVPN advertised about the keeping of records. The forum record fields are in the Discourse administrative serializer at https://raw.githubusercontent.com/discourse/discourse/main/app/serializers/admin_user_serializer.rb, read on November 11, 2026, which exposes "`ip_address`" and "`registration_ip_address`" subject to a permission check. The remaining items in this footnote are listed as possible sources rather than as recorded outcomes, and carry no case citation.

[69] `OwnerManager.sol` at release tag v1.5.0 stores owners in a linked list, "`mapping(address => address) internal owners`", and exposes "`getOwners()`". The Safe contract writes the public log entry named "`AddedOwner`" only in "`addOwnerWithThreshold`" and "`swapOwner`". `Safe.sol` at release tag v1.5.0 emits a separate entry when the Safe is created, "emit `SafeSetup(msg.sender`, _owners, _threshold, to, `fallbackHandler`);", which carries the whole initial owner list. The comment on that line states that the entry is emitted "optimistically", and that doing so makes it "easier for off-chain indexers to reliably reconstruct the Safe configuration." The function "`setupOwners`", which the "`setup`" function calls after emitting that entry, writes straight to storage and writes no further log entry. `SafeL2.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository emits the parameters of each transaction, with the nonce, the caller, and the threshold, and separately records execution by a module. The project's deployment registry lists that implementation for several blockchains at v1.4.1. The registry records which implementations are published rather than which one any particular Safe runs.

[70] That no transaction is sent from an owner blockchain address in the evasion method set out in Section 4.2, and that the owner list and the used signatures are nevertheless public, rest on the Safe contract source cited at footnote 30 and the owner records cited at footnote 69.

An owner who uses "`approveHash`" rather than signing offline must send a transaction from the owner blockchain address, and a gas fee is charged for that transaction, which the owner blockchain address had to hold the crypto-asset to meet. The proposed evasion method in Section 4.2 avoids this detection path because no owner calls "`approveHash`" at all.[71]

The reliability of blockchain tracing tools is not being assessed here because a trail of crypto-asset transfers leading to an owner blockchain address does not exist in the construction set out in Section 4.2. The blockchain does supply other starting points for tracing, including the submitter blockchain address and whatever supplied the crypto-asset for the gas fee of that blockchain address, the owner list, the signature values themselves, the sequence of nonces, and the log entry each execution writes.

Where a Safe runs the variant implementation the project publishes for blockchains other than Ethereum, the log entries carry more information, including the parameters of each transaction together with the nonce, the caller, and the threshold. Which implementation a particular Safe runs has to be checked against that Safe.[72]

The public blockchain record shows more than just keys and timing. It records the submitted instructions, the owner signatures, the submitter, and the transaction's result. Creation of the Safe and transfers supplying gas fees may provide further transaction records. These records can help an investigation, but they do not by themselves establish which person controlled each owner key or that person's knowledge and intent. Section 4.4 describes evidence that may help answer those questions. Figure 3 separates the public record from attribution to a person. Section 5 then asks whether several owner keys establish several participants. Section 6 considers the submitter's own act of execution.

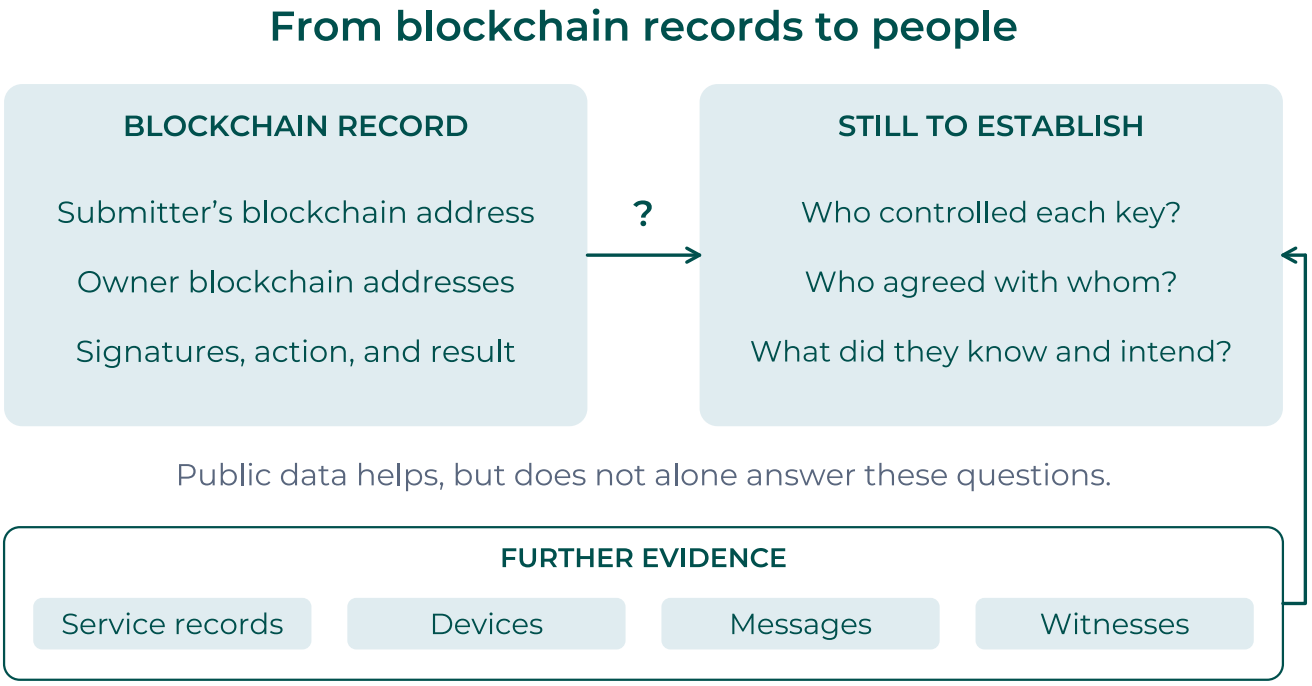


Figure 3. Public transaction data provides evidence and investigative leads. It does not alone establish who controlled each owner key, their agreement, or their knowledge and intent.[67, 69, 72]

[71] That an owner who gives an approval in advance must send a transaction from the owner blockchain address is established by the approval function cited at footnote 44. That such a transaction bears a gas fee follows from the Ethereum developer documentation cited at footnotes 31 and 32, subject to the limit recorded at footnote 30.

[72] What the blockchain does supply is established by the Safe contract source cited at footnote 30 and the owner records and variant implementation cited at footnote 69.

5. THE ANTI-SHAGGY DEFENSE

Pseudonymous governance committees also create what we believe is a new defense to certain conspiracy charges, and the defense needs a name that itself requires a short explanation.

In the year 2000, the musician Shaggy released a song called "It Wasn't Me."[73] In the song a man is caught in the act of cheating on their partner and advised to deny everything. Writing in Slate, Josh Levin applied the name "Shaggy defense" to a flat denial made in the face of strong evidence, and the phrase has been used in legal commentary since.[74] The phrase is a journalistic coinage rather than a term of art.[75]

The defense described here is the opposite. The defendant does not say "it wasn't me" but "it was all me," which is why we call it the anti-Shaggy defense.

The defense works as follows. The law requires at least two people for a conspiracy to exist.[76] The government need not know who the other people are, but it does need to prove that other people exist. Suppose every owner of a Safe is a pseudonymous blockchain address of the kind described in Section 4, and the government charges the submitter with conspiracy on the basis of carrying the owners' signatures to the blockchain. The submitter can respond that the submitter controlled every owner key. If every owner is a pseudonym for one person, and no one outside the owner list agreed with that person, there is no second person, and so there is no conspiracy as a matter of law.

Unlike the original Shaggy defense, the anti-Shaggy defense can be demonstrated, at least as to present possession of every key. The defendant can produce every private key or sign a fresh message with each one in front of the court, and the mathematics then shows that the defendant holds every key during that demonstration before the court. What the demonstration does not show however is that no one else holds a copy of the keys.[77]

The defense has a cost, because claiming sole control of every owner key is close to a claim of total control over the system, which may amount to a confession to an entirely different offense. The claim of total control also assumes that no module or other mechanism can act without the owner keys. There are nonetheless acts that are legal when done by one person and illegal when done by a group. Some financial crimes are crimes only where the conduct is carried out in the course of an ongoing business, and intent may matter as well. A defendant might therefore argue that the system was merely a test of sham governance, with no intention of carrying on a business at all.[78]

However, such a defense opens two gaps. Proving that a person holds a private key today does not prove that the same person held it at the time of the relevant acts. And proving that one person holds every owner key does not rule out an agreement with someone who holds no owner key at all. Co-conspirators could quietly hand over private keys to the defendant after the relevant acts occurred,

---

[73] The Fader, "Shaggy says his hit 'It Wasn't Me' was meant to be an 'anti-cheating' song," June 5, 2023, https://thefader.com/2023/06/05/shaggy-it-wasnt-me-anti-cheating.

[74] Josh Levin, "The Shaggy Defense: America's Favorite New Legal Term," Slate, June 7, 2010, https://slate.com/culture/2010/06/the-shaggy-defense-americas-favorite-new-legal-term.html (read November 11 and November 12, 2026). Access to that page has been intermittent, and a reader may find it temporarily unavailable. The author writes that he "dubbed this the 'Shaggy defense'" and refers to the child pornography trial of R. Kelly, which he covered for Slate in 2008. The article does not give the date on which the name was first used, so this opinion piece gives none. No later use of the phrase by any other publication is named in this opinion piece.

[75] That the phrase is a journalistic coinage is established by the article cited at footnote 74.

[76] *Rogers v. United States*, 340 U.S. 367, 375 (1951), and *Morrison v. California*, 291 U.S. 82, 92 (1934), both quoted in Section 2.

[77] That a holder of a private key can prove present control by signing a fresh message is established by the Safe documentation cited at footnote 67.

[78] That a module or other mechanism can act without the owner keys is established by the module manager source cited at footnotes 120 and 121. For the business element of the offense see 18 U.S.C. § 1960, https://www.govinfo.gov/link/uscode/18/1960, headed "Prohibition of unlicensed money transmitting businesses," which reaches a person who knowingly conducts, controls, manages, supervises, directs, or owns all or part of an "unlicensed money transmitting business." For the plurality and intent requirements of a conspiracy charge see the statute cited at footnote 12 and the model instructions cited at footnote 4.

to support the defense. Figure 4 illustrates why the number of signing keys alone cannot resolve the number of participants.[79]

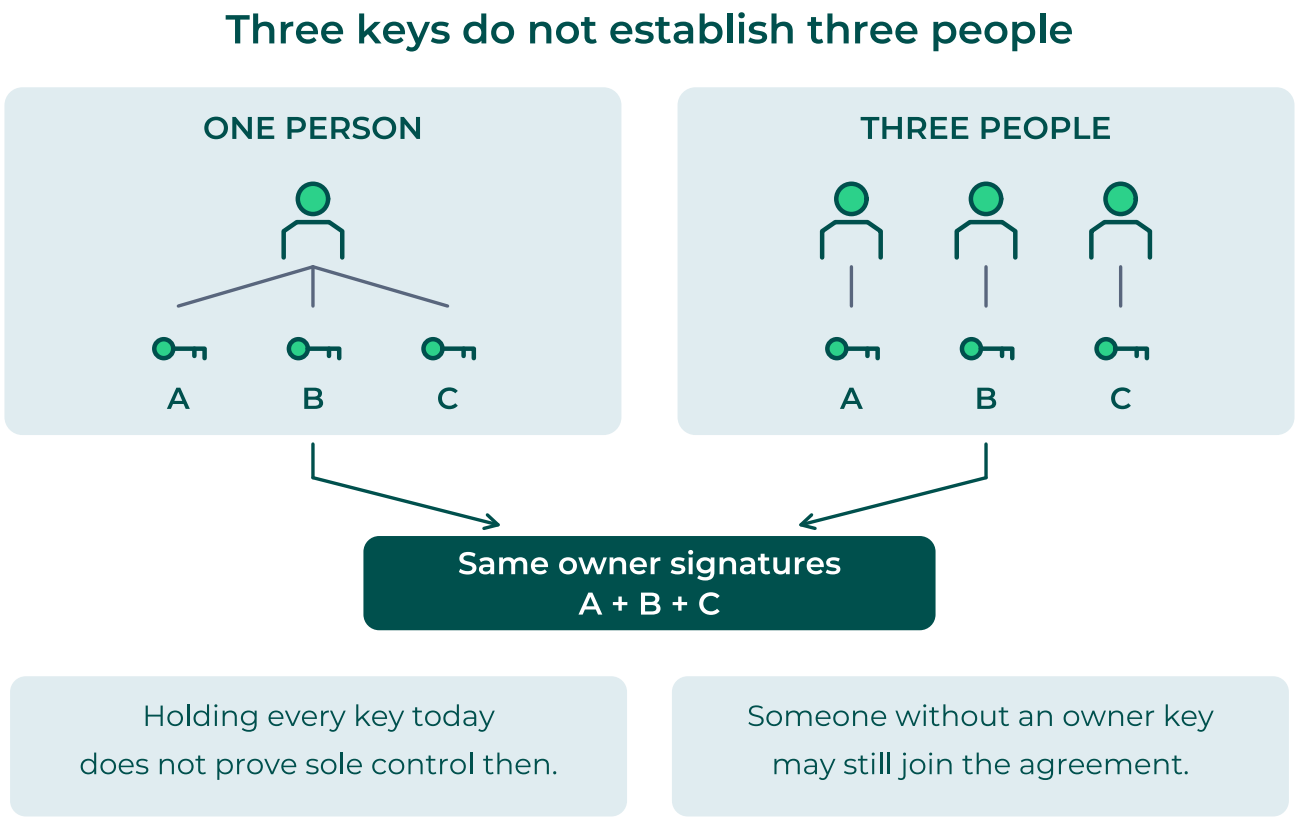


Figure 4. The same three owner signatures can be produced by one person or several. Key possession today does not establish exclusive control at the relevant time or exclude an agreement with someone outside the owner list.[76, 77, 79]

Withdrawal from a conspiracy is a related but different defense. The Supreme Court has held that a defendant who raises withdrawal in support of a statute of limitations defense carries the burden of proving it.[80] Withdrawal generally requires "[a]ffirmative acts inconsistent with the object of the conspiracy and communicated in a manner reasonably calculated to reach co-conspirators."[81] Where the statute requires an overt act, the conspiracy offense is not complete until the overt act occurs. A member who withdraws before any conspirator commits an overt act is accordingly entitled to acquittal on that charge.[82]

Two things narrow that window sharply.

The overt act may be committed by any member of the group rather than by the withdrawing member, and the overt act need not be a crime. Any lawful step by anyone in the group that furthers

[79] That a conspiracy can include a person who holds no owner key is established by the authorities cited at footnotes 20 and 21.

[80] *Smith v. United States*, 568 U.S. 106, 112 (2013) ("As with other affirmative defenses, the burden is on him."). The Court made that statement about a withdrawal claim raised in support of a statute of limitations defense. The Court also held, at 110, that "[a]llocating to a defendant the burden of proving withdrawal does not violate the Due Process Clause."

[81] *United States v. United States Gypsum Co.*, 438 U.S. 422, 464 to 465 (1978). The Court described such acts as having "generally been regarded as sufficient to establish withdrawal or abandonment."

[82] *Hyde v. United States*, 225 U.S. 347, 359 (1912) ("The conspiracy, therefore, cannot alone constitute the offense. It needs the addition of the overt act."). See also Third Circuit Model Criminal Jury Instruction 6.18.371J-1, in the chapter revised June 2026, which places the burden on the government in that setting ("If you find, based on the evidence, that (name) withdrew from the conspiracy before any conspirator committed any overt act, then you must find (name) not guilty of conspiracy").

the object of the conspiracy, taken before the withdrawal reaches the others, closes the window. The burden of proof in that situation is also unsettled.

The Supreme Court decided the burden question in the setting of a limitations defense and did not address the overt act element. There is a reasonable argument that withdrawal before any overt act negates an element of the offense rather than operating as an affirmative defense, which would place the burden on the government.[83] After the offense is complete, withdrawal does not undo the defendant's own liability for the conspiracy, although it cuts off liability for what the others do afterward and can start the limitations clock running.

The Supreme Court has put the rule plainly. "Withdrawal terminates the defendant's liability for postwithdrawal acts of his co-conspirators, but he remains guilty of conspiracy."[84] Handing over a private key, or publicly giving up a private key, could be solid evidence of an affirmative act of withdrawal communicated to the group. The act must be inconsistent with the object of the conspiracy rather than a step that helps the conspiracy continue. Giving up one copy of a key does not establish that no copy remains.[85]

There are lessons here for developers, since a developer who may need to prove sole control, or to prove withdrawal before any conspirator committed an overt act, should preserve the evidence establishing it.

There are lessons for law enforcement as well. An investigator who wants to defeat the anti-Shaggy defense should collect evidence that different pseudonyms were controlled by different people at the time the relevant acts occurred, rather than merely evidence of who holds the private keys later.

6. PRACTICAL RESULTS

Section 4 provided a proposed evasion method for signing multisig transactions under which no owner sends a transaction at any point. No transfer of crypto-assets reaches an owner blockchain address, so the trail that blockchain tracing typically follows is absent. The one party whose own transaction carries out a Safe transaction is the submitter who carries the signatures there.[86]

Now assume the Safe is facilitating a criminal act. As Section 1 says, this opinion piece ignores both jurisdiction and the question of which statute applies. Assume only that the Safe is running a mixing service and that the way the Safe runs that mixing service is illegal. A mixing service receives crypto-assets from many people and sends crypto-assets back out in a way that breaks the link on the blockchain between what came in and what went out.[87]

Who can be charged?

An owner could be charged if the government proves participation in the criminal agreement with the required knowledge and intent. Attributing an owner blockchain address to that person is a separate evidential task. Blockchain tracing has no transfer trail to follow, because no transfer of crypto-assets reaches an owner blockchain address. This opinion piece does not assess what any tracing method can do from other starting points. Whether an investigation could reach the owners by other means, such

---

[83] The burden where withdrawal is said to precede any overt act was not decided in *Smith v. United States*, 568 U.S. 106 (2013), which concerned a withdrawal claim raised in support of a statute of limitations defense. The formulation in the Third Circuit model instruction on this point has changed between editions, so any reliance should identify and quote the edition consulted. The June 2026 edition of that chapter was read for this opinion piece. The 2021 edition of the same chapter is published at https://www.ca3.uscourts.gov/sites/ca3/files/2021%20Chapter%206%20Conspiracy%20for%20posting%20final.pdf. That edition was not obtained for this opinion piece, so a reader comparing the two editions should consult it directly.

[84] *Smith v. United States*, 568 U.S. 106, 111 (2013).

[85] That the act relied on must be inconsistent with the object of the conspiracy is established by the authority cited at footnote 81.

[86] That no owner sends a transaction in the evasion method set out in Section 4, and that the submitter's transaction is the one that carries out the Safe transaction, rest on the Safe contract source cited at footnote 30 and the owner records cited at footnote 69.

[87] The description of a mixing service in this paragraph is the opinion piece's own definition, offered for the hypothetical, and no source is cited for it.

as the signatures themselves, the forum posts, service records, devices, witnesses, or the submitter, is a separate question that this opinion piece does not answer. Whether an investigation could attribute single-use messages posted to public forums to particular people is a question about investigative resources and about the records third parties keep. That question is not one the blockchain answers, and this opinion piece does not attempt to answer it either.[88]

The one party whose own transaction carries out a Safe transaction is the submitter. We are not arguing that the submitter should have to disclose who they are to anyone, that blockchains should be permissioned, or that the submitter should be forced to make tracing easier. Our argument is only that the submitter is a valid enforcement target, for two reasons.[89]

First, the submitter changes the system by sending the deployment transaction. Our argument is that, where the submitter knowingly joined the criminal agreement and intended to further its object, executing the agreed change can constitute an overt act in furtherance of that agreement. The overt act need not itself be criminal.[90] Where no overt act is required, the transaction can still be evidence bearing on participation in the agreement. Neither proposition means that sending a transaction alone proves agreement, knowledge, or intent.

Second, the submitter is the only person whose own transaction records the furtherance. The owners furthered the conspiracy by signing, and those signatures become public when the transaction executes, but no owner sent a transaction. The submitter's transaction is a permanent public entry made from a blockchain address the submitter controlled.[91]

We are not arguing that the submitter should be accountable merely because there is no one else to go after, although whether anyone else joined the agreement is nonetheless relevant to the overall question. The submitter certainly did something that one can reasonably argue was in furtherance of the conspiracy. The case described here is not one of charging the developers because there are no operators, because the submitter *is* an operator.

One objection should be addressed directly, which is that submitting signatures is merely sending a message, and that sending a message is speech. The Supreme Court rejected that form of argument long ago. "[I]t has never been deemed an abridgment of freedom of speech or press to make a course of conduct illegal merely because the conduct was in part initiated, evidenced, or carried out by means of language, either spoken, written, or printed."[92] The same argument was made and rejected in the Tornado Cash prosecution. Judge Katherine Polk Failla of the Southern District of New York denied the motion to dismiss in an oral ruling on September 26, 2024. Judge Failla said that "these laws do not target protective [sic] expressive conduct" and that the charged conduct "implicates the functional, rather than the expressive features, of computer coding and software."[93] Submitting signed votes to effect a deployment is conduct. That the conduct is carried out by means of data does not by itself put it beyond the reach of a criminal statute. Whether a particular prosecution offends the First Amendment is a separate question, which turns on the statute and on the conduct charged, and this opinion piece does not deal with that situation.[94]

---

88 That no transfer of crypto-assets reaches an owner blockchain address in the evasion method set out in Section 4.2 rests on the Safe contract source cited at footnote 30 and the owner records cited at footnote 69.

89 That the submitter's transaction is the one that carries out the Safe transaction rests on the Safe contract source cited at footnotes 30 and 31.

90 *Yates v. United States*, 354 U.S. 298, 334 (1957), and *Braverman v. United States*, 317 U.S. 49, 53 (1942), both quoted in Section 2.

91 That the owner signatures become public when the transaction executes, and that no owner sends a transaction, rest on the Safe contract source cited at footnotes 30, 31, and 38.

92 *Giboney v. Empire Storage & Ice Co.*, 336 U.S. 490, 502 (1949). The Court also observed, at 498, that "[i]t rarely has been suggested that the constitutional freedom for speech and press extends its immunity to speech or writing used as an integral part of conduct in violation of a valid criminal statute."

93 Transcript of oral decision, *United States v. Storm*, No. 23 Cr. 430 (KPF) (S.D.N.Y. Sept. 26, 2024), Dkt. 84, filed October 3, 2024. The ruling was delivered orally and transcribed. The word "protective" appears in the transcript as filed and is quoted here as transcribed. See also André Beganski, "Why Tornado Cash Co-Founder Roman Storm's Free Speech Defense Was Rejected in Court," Decrypt, September 30, 2024, https://decrypt.co/283890/tornado-cash-roman-storm-free-speech-defense. Judge Failla denied reconsideration on February 19, 2025. Order, Dkt. 127, filed and dated February 19, 2025, four pages, denying reconsideration as to all three counts.

94 That the conduct charged in the Tornado Cash prosecution was treated as conduct rather than speech is established by the transcript cited at footnote 93. The general distinction is at footnote 92.

On the blockchain record alone, the submitter is the only party whose own transaction carries out a Safe transaction, because the owners made none. The signatures that transaction carries show which keys authorized the change, and attributing those keys to people requires evidence from outside the blockchain. That observation is about what information the blockchain provides and not about every route an investigation might take. The contention of this opinion piece is that the transaction submitting the signatures can support prosecution where the government can also prove the submitter's knowing participation in the criminal agreement and intent to further its object.[95]

7. WHERE THE LAW AND POLICY STAND TODAY

The argument in Section 6 does not exist in a vacuum, and three developments bear on it.

7.1 The Tornado Cash prosecution

A United States criminal trial has reached a verdict against a developer of Tornado Cash, software used to mix crypto-assets. That trial, and the only one this opinion piece has identified, is *United States v. Storm*, No. 1:23-cr-430 (S.D.N.Y.).[96] Roman Storm was indicted in August 2023 on three conspiracy counts. The counts were conspiracy to commit money laundering, conspiracy to operate an unlicensed money transmitting business, and conspiracy to violate sanctions law. Because § 1960 contains no conspiracy provision of its own, the second count was charged under the general conspiracy statute, making its precise form a conspiracy under 18 U.S.C. § 371 to violate 18 U.S.C. § 1960(a) and (b)(1)(C).[97] On August 6, 2025, the jury convicted Mr. Storm on that second count. The jury could not agree on the other two counts.[98] One point about the second count bears on Section 7.2 below. As originally charged, the count rested on both 18 U.S.C. § 1960(b)(1)(B), which concerns failure to register with the Financial Crimes Enforcement Network, and § 1960(b)(1)(C), which concerns transmitting criminally derived property. On May 15, 2025 the government abandoned the subsection (b)(1)(B) theory and tried the case on subsection (b)(1)(C) alone.[99]

Two points follow from this verdict. First, the one count of conviction was a § 371 conspiracy, and § 371 requires an overt act. The overt act question in this opinion piece is therefore live in that case. Second, the two counts on which the jury could not agree are being retried, with the retrial set for April 26, 2027. Mr. Storm's motion for a judgment of acquittal on the count of conviction was argued on April 9, 2026.

---

[95] That the signatures show which keys authorized the change, and that attributing those keys to people needs evidence from outside the blockchain, rest on the Safe documentation cited at footnote 67 and the record of attribution cited at footnote 68.

[96] *United States v. Storm*, No. 1:23-cr-430 (KPF) (S.D.N.Y.). The docket number and the assignment are recorded in the filed documents cited in this subsection. This opinion piece has identified no earlier United States criminal trial of a developer of decentralized finance software that reached a verdict. No systematic survey of such prosecutions was carried out, so the claim is limited to what this opinion piece identified.

[97] Count Two of the indictment charged a conspiracy under 18 U.S.C. § 371 to violate 18 U.S.C. § 1960. The statutory provisions tried are recorded in the brief of Paradigm Operations LP as amicus curiae, *United States v. Storm*, No. 1:23-cr-430 (KPF) (S.D.N.Y.), Dkt. 166-1, filed June 13, 2025, at printed page 4, and in the defense letter at Dkt. 306, filed October 9, 2026. Those filings are the parties' descriptions and are not the operative charging instrument or the jury instructions. For the original charging instrument see the indictment in that prosecution, unsealed in August 2023 in the Southern District of New York and published by the Department of Justice at https://www.justice.gov/d9/2023-08/u.s._v._storm_and_semenov_indictment.pdf, together with the Department announcement of August 23, 2023 at https://www.justice.gov/archives/opa/pr/tornado-cash-founders-charged-money-laundering-and-sanctions-violations. The announcement, read on November 10, 2026, is numbered 23-907, is headed "Tornado Cash Founders Charged with Money Laundering and Sanctions Violations," records that the indictment was "unsealed today in the Southern District of New York," and states that "Storm and Semenov are each charged with one count of conspiracy to commit money laundering." It describes three charges against each defendant and gives no statutory citation for any of them, so the statutory provisions tried rest on the filings cited above rather than on the announcement. The indictment itself is published as a scanned image carrying no text layer, and it was not quoted for this opinion piece.

[98] Verdict form, *United States v. Storm*, No. 1:23-cr-430 (KPF) (S.D.N.Y.), Dkt. 224, filed August 7, 2025, and order, Dkt. 303, filed September 10, 2026, which records the verdict as returned on August 6, 2025 and records the partial verdict. The filing date of the verdict form is not the date of the verdict. A jury that could not agree on a count has not acquitted on it.

[99] Letter of the United States, *United States v. Storm*, No. 1:23-cr-430 (KPF) (S.D.N.Y.), Dkt. 144, filed May 15, 2025, in which the government states that it will not proceed on the registration theory and will continue on the other identified theories. The letter is the operative record of the narrowing.

The latest filing obtained for this opinion piece, dated October 9, 2026, shows that motion still pending and no later docket entry was obtained.[100]

7.2 Department of Justice policy

On April 7, 2025, Todd Blanche, then Deputy Attorney General, issued a memorandum to all Department of Justice employees titled "Ending Regulation By Prosecution."[101] Mr. Blanche has since become Attorney General, having been sworn in on August 10, 2026. A search of public Department of Justice sources for a rescission or modification of the memorandum returned none on October 30, 2026. The memorandum directs the Department to "no longer target virtual currency exchanges, mixing and tumbling services, and offline wallets for the acts of their end users or unwitting violations of regulations" except where an investigation is "consistent with the priorities articulated in the following paragraphs." The memorandum also directs that regulatory charges, including unlicensed money transmitting, not be brought "unless there is evidence that the defendant knew of the licensing or registration requirement at issue and violated such a requirement willfully."[102]

One limit on the memorandum bears directly on the Tornado Cash prosecution. The charging restriction is directed at subsections (b)(1)(A) and (b)(1)(B) of 18 U.S.C. § 1960, which concern licensing and registration. The memorandum says in a footnote that subsection (b)(1)(C) imposes a condition, in words the memorandum quotes from the statute, that the property transmitted "are known to the defendant to have been derived from a criminal offense or are intended to be used to promote or support unlawful activity," and is "therefore outside the scope of this policy."[103] Subsection (b)(1)(C) therefore reaches property of two kinds, and it is the subsection Mr. Storm was convicted of conspiring to violate. The memorandum did not stop that prosecution.

On August 21, 2025, Matthew Galeotti, then Acting Assistant Attorney General and head of the Criminal Division, said in a speech that "merely writing code, without ill-intent, is not a crime." Mr. Galeotti then set a condition in two parts. Where "the evidence shows that software is truly decentralized and solely automates peer-to-peer transactions, and where a third party does not have custody and control over user assets," new charges under subsection (b)(1)(C) against that third party "will not be approved." Mr. Galeotti added that "if criminal intent is present, other charges may be appropriate," and that all of a subject's "conduct and the services they provide end-to-end will be considered."[104]

---

[100] Order, *United States v. Storm*, No. 1:23-cr-430 (KPF) (S.D.N.Y.), Dkt. 300, filed August 25, 2026, setting the retrial for April 26, 2027. Order, Dkt. 303, filed September 10, 2026, records that the motion for a judgment of acquittal was argued on April 9, 2026. The defense letter at Dkt. 306, filed October 9, 2026, still concerns the pending motion. No complete current docket was obtained, so the status of the motion after October 9, 2026 is not established here, and the retrial date can change.

[101] Memorandum from the Deputy Attorney General, "Ending Regulation By Prosecution," April 7, 2025, https://www.justice.gov/dag/media/1395781/dl?inline= (accessed October 30, 2026). The subject line of the memorandum capitalizes "By". For the swearing-in date see the Attorney General staff profile on the Department of Justice website. A search of Department of Justice sources on October 30, 2026 returned no rescission or modification of the memorandum.

[102] Ending Regulation By Prosecution, April 7, 2025. For the swearing-in date in the preceding sentence see the Attorney General staff profile cited at footnote 101. The no-targeting direction and its exception sit in the section headed "Digital Assets Enforcement Priorities". The charging direction sits in the section headed "Digital Assets Charging Considerations". The memorandum records that the willfulness condition is "not required by law, but is being imposed as a matter of discretion . . . ."

[103] Ending Regulation By Prosecution, April 7, 2025, footnote 2, which reads that "18 U.S.C. § 1960(b)(1)(C) requires that the transmission of funds 'are known to the defendant to have been derived from a criminal offense or are intended to be used to promote or support unlawful activity,' and is therefore outside the scope of this policy." A search of the memorandum for the word "conspiracy" on October 30, 2026 returned no occurrence. The memorandum is dated before the verdict and before the narrowing of the charge in that prosecution, and cannot establish either. For those case-specific matters see the records cited at footnotes 98 and 99.

[104] Remarks of Acting Assistant Attorney General Matthew R. Galeotti at the American Innovation Project Summit, Jackson, Wyoming, August 21, 2025, https://www.justice.gov/opa/speech/acting-assistant-attorney-general-matthew-r-galeotti-delivers-remarks-american (accessed October 30, 2026). The remarks do not mention the Tornado Cash prosecution. The passage on mental state reads that "[t]o be guilty of aiding and abetting a crime, one has to intend to aid the commission of an underlying crime. It requires specific intent. So does conspiracy." The source sets the phrase "or who aid and abet the commission of crimes" between dashes, which this opinion piece does not reproduce, so the continuing enforcement statement is not quoted here. For secondary coverage see Sarah Wynn, "'Writing code without ill intent is not a crime,' says US DOJ official following Tornado Cash verdict," The Block, August 21, 2025, https://www.theblock.co/post/367858/writing-code-without-ill-intent-is-not-a-crime-says-us-doj-official-following-tornado-cash-verdict, whose headline normalizes the punctuation of the quoted phrase.

The same speech addressed conspiracy directly. Mr. Galeotti said that aiding and abetting "requires specific intent," and that "[s]o does conspiracy." Mr. Galeotti said that a developer who "merely contributes code to an open-source project, without the specific intent to assist criminal conduct, aid or abet a crime, or join a criminal conspiracy," is "not criminally liable."[105]

Mr. Galeotti had left the Department of Justice by March 2026,[106] and his remarks stand as a statement of the position the Criminal Division took at the time. However, this opinion piece does not assert that Galeotti's position has been reaffirmed since.

Neither statement forecloses a charge against the submitter, and neither resolves one.

The submitter does more than write code, because the submitter sends the transaction that changes a system in operation. Whether the protection described in Galeotti's speech would cover such a submitter turns on the two conditions his speech sets. The first is whether the software is truly decentralized and solely automates transactions between users, and the second is whether the submitter has custody and control of user assets. Both are questions of fact about a particular system and a particular person.[107]

Galeotti's speech states the Criminal Division's position at the time that conspiracy requires specific intent. It does not decide whether a particular deployment transaction proves agreement, satisfies an overt act requirement, or establishes any other element. This opinion piece infers from Galeotti's position that the state of mind of the submitter is where the argument will be contested, and the inference is this opinion piece's rather than the Department's. The question this opinion piece raises is therefore whether sending the deployment transaction is evidence from which such intent can be found. A search of the memorandum for the word "conspiracy" returned nothing, so the memorandum at least provides no specific disposition for a deployment transaction. Directions it gives about investigations and charges could still bear on a conspiracy prosecution.[108]

7.3 Legislation

On January 12, 2026, Senator Cynthia Lummis, joined by Senator Ron Wyden, introduced the Blockchain Regulatory Certainty Act of 2026, S. 3611.[109] The bill provides that a "non-controlling developer or provider" shall not be treated as "a money transmitting business" under 31 U.S.C. § 5330 or as "engaged in money transmitting" under 18 U.S.C. § 1960. The bill defines a non-controlling developer or provider as "a developer or provider of a distributed ledger service that, in the regular course of operations, does not have the legal right or the unilateral and independent ability to control, initiate upon demand, or effectuate transactions involving digital assets to which users are entitled, without the approval, consent, or direction of any other third party." The qualifier "in the regular course of operations" is part of the definition and narrows it.[110]

The bill covers money transmitter status, but contains no provision about conspiracy. The absence of such a provision is not the same as having no effect on a conspiracy charge. The second *Storm* count

---

[105] Galeotti remarks, August 21, 2025, cited in the preceding footnote, where the mental state passage is quoted in full. The developer passage reads that a person who "merely contributes code to an open-source project, without the specific intent to assist criminal conduct, aid or abet a crime, or join a criminal conspiracy," is "not criminally liable." The bracketed initial lowercase in the text is an alteration made for this opinion piece.

[106] Kirkland & Ellis, "Matthew Galeotti, Former Acting Assistant Attorney General of DOJ's Criminal Division, Joins Kirkland & Ellis," March 9, 2026, https://www.kirkland.com/news/press-release/2026/03/matthew-galeotti-former-aaag-of-dojs-criminal-division-joins-ke. The release establishes that Mr. Galeotti had left the Department of Justice by that date. The release does not give the date of Mr. Galeotti's departure.

[107] The two statements described in this paragraph are quoted from the memorandum cited at footnotes 101 to 103 and from the remarks cited at footnotes 104 and 105.

[108] The stated position of the Criminal Division on specific intent is quoted from the remarks cited at footnotes 104 and 105. For the memorandum, the scope of its directions, and the search of the memorandum for the word "conspiracy", see footnotes 101 to 103.

[109] S. 3611, 119th Cong., 2d Sess. (introduced January 12, 2026), https://www.govinfo.gov/content/pkg/BILLS-119s3611is/html/BILLS-119s3611is.htm. See also Office of Senator Cynthia Lummis, "Lummis, Wyden Introduce Bipartisan Legislation to Protect Blockchain Developers from Money Transmitter Requirements," January 12, 2026, https://www.lummis.senate.gov/press-releases/lummis-wyden-introduce-bipartisan-legislation-to-protect-blockchain-developers-from-money-transmitter-requirements.

[110] S. 3611, § 2(a)(5) and § 2(b).

was a conspiracy to violate § 1960, so a bill that narrows who falls within § 1960 can narrow the object of such a conspiracy. Whether it would do so for any given defendant turns on the bill's operative and construction provisions, which this opinion piece does not analyze. The bill was referred to the Senate Committee on Banking, Housing, and Urban Affairs and has not become law.[111] The substance of the bill was taken up in the Senate market structure negotiations. The Senate Banking Committee advanced its market structure bill by a vote of 15 to 9 on May 14, 2026. The text the Committee advanced carried the substance of the Blockchain Regulatory Certainty Act, excluding non-custodial developers and providers from the money transmitter definitions in 31 U.S.C. § 5330 and 18 U.S.C. § 1960.[112] A proposed substitute was released on September 14, 2026, which the sponsors said would be offered if the Senate agreed to proceed to the bill.[113] That substitute was not adopted, because the motion to invoke cloture on the motion to proceed failed on September 15, 2026, and the official action history records no later adoption of it.[114]

The House passed the Digital Asset Market Clarity Act of 2025, H.R. 3633, on July 17, 2025, by 294 to 134. On September 15, 2026, the Senate rejected the cloture motion needed to proceed to debate on that bill. The vote was 49 to 50, short of the 60 votes required. Senator Thom Tillis entered a motion to reconsider, preserving a procedural route back to the floor. The official action history records no later floor action as of November 14, 2026.[115]

Congress has considered protecting developers who do not control transactions, but it has not enacted the protection examined here. The protection in the text the Committee reported was itself limited. It preserved the application of subsection (b)(1)(C) of 18 U.S.C. § 1960 to conduct undertaken with specific

---

111 S. 3611, header ("[r]ead twice and referred to the Committee on Banking, Housing, and Urban Affairs"). The header establishes the referral rather than the later history of the bill. The official bill status record for S. 3611, https://www.govinfo.gov/bulkdata/BILLSTATUS/119/s/BILLSTATUS-119s3611.xml, returned the referral of January 12, 2026 as the latest action when it was read on November 10, 2026, and checked again on November 14, 2026. For the charge brought in the Tornado Cash prosecution and its narrowing see the records cited at footnotes 97 and 99.

112 Stephen T. Gannon and Elizabeth Lan Davis, "Senate Banking Committee Advances Crypto Market Structure Bill," Davis Wright Tremaine, May 21, 2026, https://www.dwt.com/blogs/financial-services-law-advisor/2026/05/senate-banking-crypto-market-structure-bill ("Developer protections that are consistent with a 'regulatory certainty' approach for noncontrolling software developers and providers are included in the substitute."). The Senate reported text of H.R. 3633 dated June 1, 2026, https://www.govinfo.gov/content/pkg/BILLS-119hr3633rs/html/BILLS-119hr3633rs.htm, is the primary citation for the text as advanced, and it was read in full on November 10, 2026. Section 604(c)(1) provides that a non-controlling developer or provider "shall not be treated as" either "a money transmitting business, as defined in section 5330 of title 31, United States Code" or "engaged in money transmitting, as defined in section 1960 of title 18, United States Code." Section 604(d) provides that subsection (c) "shall not modify the application of section 1960(b)(1)(C) of title 18, United States Code" to a person who acts with the specific intent described in that subsection. The inner quotation marks around the phrase regulatory certainty are reproduced as typographic marks, and the source renders them as plain marks.

113 For the Committee text as advanced, see Senate Committee on Banking, Housing, and Urban Affairs, "Chairman Scott, Senate Banking Committee Advance CLARITY Act in Historic Bipartisan Vote," May 14, 2026, https://www.banking.senate.gov/newsroom/majority/chairman-scott-senate-banking-committee-advance-clarity-act-in-historic-bipartisan-vote. The official announcement of the proposed substitute is dated September 14, 2026, and the September 13 circulation date is not established from a dated primary record. This opinion piece does not state what the proposed substitute of September 14, 2026 contained. Press reporting of that date described the released text as removing the criminal protections of the Blockchain Regulatory Certainty Act. The article relied on in earlier revisions of this opinion piece could not be retrieved for this revision and its displayed title was not confirmed, so it is not cited here. The draft carrying the September 14 text was not obtained. A different draft, an amendment in the nature of a substitute to H.R. 3633 numbered EHF26654 FYM, was released by Senator Lummis on July 22, 2026 and published at https://www.lummis.senate.gov/wp-content/uploads/Clarity-Act.pdf. For the release see Office of Senator Cynthia Lummis, "Lummis Releases Updated Clarity Act Text," July 22, 2026, https://www.lummis.senate.gov/press-releases/lummis-releases-updated-clarity-act-text/, read on November 11, 2026. That draft carries the same renumbering of the section to 10604 and so predates September 14, 2026. That draft was read in full on November 10, 2026 from a copy bearing the identifier EHF26654 FYM in its page headers, and it retains the express exclusion at § 10604(c)(1)(B), at printed page 310, in terms identical to § 604(c)(1)(B) of the reported text quoted at footnote 112. The subsection (d) qualification is retained at printed page 311. The contents of the September 14 text are therefore left unstated, and no part of this opinion piece turns on them.

114 That the text released on September 14, 2026 was a proposed substitute, which the sponsors said would be offered if the Senate agreed to proceed, is established by the announcement of that date at https://www.lummis.senate.gov/press-releases/lummis-boozman-scott-release-final-clarity-act-text/, read on November 10, 2026, which states that "[i]f cloture is invoked on the motion to proceed on Tuesday afternoon, this legislative text would be offered as an Amendment in the Nature of a Substitute." That the substitute was not adopted is established by the failed cloture vote and the official action history recorded at footnote 115.

115 Office of the Clerk, United States House of Representatives, Roll Call 199, July 17, 2025, recording passage by 294 to 134. United States Senate, Roll Call 234, September 15, 2026, recording 49 to 50 on the motion to invoke cloture on the motion to proceed to H.R. 3633, which required three fifths of the Senate. That vote was on cloture and was not a final vote rejecting passage. Senate floor activity for September 15, 2026 records the motion to reconsider entered by Senator Thom Tillis. September 15, 2026 is the latest action returned by the official bill status record for H.R. 3633, https://www.govinfo.gov/bulkdata/BILLSTATUS/119/hr/BILLSTATUS-119hr3633.xml, when it was read on November 10, 2026, and checked again on November 14, 2026. That record shows no adoption of the proposed substitute released on September 14, 2026. The Senate roll call is at https://www.senate.gov/legislative/LIS/roll_call_votes/vote1192/vote_119_2_00234.htm, which records the question as cloture on the motion to proceed and the result as rejected.

intent and with knowledge concerning criminally derived property or an intended unlawful use. That text therefore already left criminal conduct of that kind outside its protection. Neither S. 3611 nor H.R. 3633, which are the measures examined for this opinion piece, says that a person who sends a transaction changing a system that facilitates criminal activity is exempt from conspiracy law. [116]

8. CONCLUSION

The proposed evasion method in Section 4 uses only standard features of the Safe, and no owner sends a transaction at any point. The only transaction that carries out a Safe transaction is the submitter's.[117]

If the submitter's transaction counts neither as an overt act where an overt act is required, nor as evidence of joining the agreement where none is required, then no transaction that carries out a Safe transaction supplies conduct the government can charge against anyone who runs a criminal system in this manner. The published owner signatures remain available as evidence, and Section 4.5 sets out what they do and do not show. That conclusion is about the act and the evidence of agreement. That conclusion assumes, and does not supply, the knowledge and intent the offense requires.[118]

If the law is supposed to be that transactions sent through a blockchain are exempt from conspiracy law, Congress should change the law to say so. Until Congress is willing to provide such broad and unprecedented protection in express terms, there is no justification for reading one into the statutes.

[THIS SPACE INTENTIONALLY LEFT BLANK]

[116] That Congress has considered and not enacted the protection described is established by the bill cited at footnotes 109 to 111, the reported text quoted at footnote 112, and the votes recorded at footnote 115. That the text released on September 14, 2026 was a proposed substitute rather than an adopted amendment is established by the announcement and the official records recorded at footnotes 114 and 115.

[117] That the evasion method uses only standard features of the Safe, and that no owner sends a transaction, rest on the Safe contract source cited at footnote 30 and the Safe documentation cited at footnote 25.

[118] That the published owner signatures remain available as evidence rests on the Safe contract source cited at footnotes 30 and 38.

Appendix. Two further ways a Safe can act

Section 3 describes the ordinary approval process, in which enough owners sign a Safe transaction and a submitter calls "`execTransaction`". A Safe can be made to act in two further ways. The first requires no owner signature at all when the Safe acts. On the ordinary transaction route, the second still requires the full number of signatures the threshold demands, but it lets a single approved Safe transaction rewrite the rules of the Safe itself. Both matter to the question of who can be charged with criminal conspiracy, because each one changes what the system does, and each one still records a caller on the blockchain.[119]

A module is a blockchain address that the Safe has enabled to act for it. It can be a smart contract or a blockchain address controlled by a private key. Enabling it checks that the blockchain address is neither empty nor the special marker ending the module list, and that it is not already enabled. The Safe contract comments describe a module as an extension "with unlimited access to a Safe" and warn that "[a] malicious module can completely takeover a Safe."[120] An enabled module uses "`execTransactionFromModule`", which does not require a new set of owner signatures or check the owner threshold. Version 1.5.0 also permits an optional module guard, which checks before and after the module acts and can refuse the transaction.[121]

Owners can enable a module through an ordinary Safe transaction that meets the current threshold. The "`enableModule`" function itself checks that the caller is the Safe, rather than checking owner signatures. A module can therefore also be enabled when the Safe is created or by another enabled module.[122] Where owners approve a module, they authorize it to act later without fresh signatures for each action. But an enabled module does not by itself prove that the listed owners approved it. The creation process can install an owner list without checking the intended owners' consent or possession of the keys. An existing module can also enable another module the owners never directly approved.

The second route is via a delegate call, which is an operation a Safe transaction can specify rather than a separate party. Every Safe transaction carries a field named "`operation`", where the value 0 means an ordinary call and the value 1 means a delegate call.[123] A delegate call runs another smart contract code against the stored data of the Safe itself. On the ordinary route a delegate call is approved like any other Safe transaction, because the signature check runs before the Safe carries out the call. On the other two routes a delegate call is not approved that way. A module may ask for a delegate call, and the module route runs no signature check and no threshold check. The Safe may also perform a delegate call at the moment the Safe is created, before any owner has signed anything, and the Safe does so where the party creating the Safe names a destination for that call. What a single approved delegate

---

[119] The two further routes, and the operation field that selects an ordinary call or a delegate call, are in the module manager source cited at footnote 120 and the operation values cited at footnote 123. The ordinary threshold check, and the ordinary route to a delegate call, are in `Safe.sol`, cited at footnotes 30 and 31.

[120] `ModuleManager.sol` at release tag v1.5.0 in the safe-global/safe-smart-account repository, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/base/ModuleManager.sol (accessed October 27, 2026). The full comment reads "Modules are extensions with unlimited access to a Safe that can be added to a Safe by its owners. WARNING: Modules are a security risk since they can execute arbitrary transactions, so only trusted and audited modules should be added to a Safe. A malicious module can completely takeover a Safe." The comment is reformatted onto one line, and the warning marker that precedes the word WARNING in the source is omitted here.

[121] `ModuleManager.sol` at release tag v1.5.0. The check sits in "`preModuleExecution`" and reads "`if (msg.sender == SENTINEL_MODULES || modules[msg.sender] == address(0)) revertWithError("GS104");`" under the comment "Only whitelisted modules are allowed." Neither "`execTransactionFromModule`" nor "`execTransactionFromModuleReturnData`" carries the "`authorized`" requirement, and neither contains a signature or threshold check.

[122] `ModuleManager.sol` at release tag v1.5.0. The declaration reads "`function enableModule(address module) public override authorized`". The Safe contract writes a public log entry named "`EnabledModule`" when a module is enabled and "`DisabledModule`" when a module is disabled, and exposes "`getModulesPaginated`" and "`isModuleEnabled`" so that the enabled modules can be read. The self-call test is in the source cited at footnote 28, the ordinary threshold check in the source cited at footnote 30, and the owner list installed when the Safe is created in the source cited at footnote 27. That no consent of a named owner is required at creation follows from those same sources.

[123] `Enum.sol` at release tag v1.5.0, https://raw.githubusercontent.com/safe-global/safe-smart-account/v1.5.0/contracts/libraries/Enum.sol (accessed October 27, 2026), declares "`enum Operation { Call, DelegateCall }`". Solidity assigns the values in declaration order, so "`Call`" is 0 and "`DelegateCall`" is 1. `Executor.sol` at the same release tag dispatches on the value. For the point in the ordinary path at which signatures are checked, and for the module path, see `Safe.sol` cited at footnotes 30 and 31 and `ModuleManager.sol` cited at footnotes 120 and 121. Each path can reach a delegate call, because the operation field is carried through on both. Code quotations in this footnote are reformatted onto one line from the source.

call can do is rewrite the owner list and the threshold directly, without calling the owner management functions at all. The threshold constrains each transaction, but the threshold is not a durable constraint against a single approved transaction that removes the threshold.

Neither route changes the conclusion in Section 6, because each still requires that someone send a transaction. On the module route the caller the Safe sees is the enabled module, which acts by calling "`execTransactionFromModule`". Where that module is itself a smart contract, a separate blockchain address must send the transaction that reaches it. Where the enabled module is an ordinary blockchain address controlled by a private key, that same party sends the transaction and no second party is required.

A delegate call can be reached on any of the three routes. At setup the sender is the party creating the Safe. On the module route the sender is the blockchain address that reached the module, which is the module itself where the module is an ordinary blockchain address and a separate blockchain address where the module is a smart contract. On the ordinary route the sender is the submitter who calls "`execTransaction`" in the ordinary way.

In each of these routes at least one party appears on the blockchain as having advanced the system. The evasion method in Section 4.2 is the case in which exactly one party appears as the sender of a transaction advancing the system, and that party is the one this opinion piece argues is a valid enforcement target. The main text considers the ordinary signature route, because that is the route on which no owner sends a transaction of any kind. Creating the Safe was itself a transaction with a sender, so the party who advanced the system is never the only party the blockchain records across the life of the Safe.[124]

[124] The caller the Safe sees on each route, and the party who sends the transaction that reaches it, are established by the Safe contract source cited at footnote 30, the module manager source cited at footnotes 120 and 121, and the operation values cited at footnote 123.