Risk Framework for Bitcoin Custody Operation with the Revault Protocol

Jacob Swambo[1,2] and Antoine Poinsot[2]

[1] King's College London, Department of Informatics
[2] WizardSardine

Abstract. Our contributions with this paper are twofold. First, we elucidate the methodological requirements for a risk framework of custodial operations and argue for the value of this type of risk model as complementary with cryptographic and blockchain security models. Second, we present a risk model in the form of a library of attack-trees for Revault – an open-source custody protocol. The model can be used by organisations as a risk quantification framework for a thorough security analysis in their specific deployment context. Our work exemplifies an approach that can be used independent of which custody protocol is being considered, including complex protocols with multiple stakeholders and active defence infrastructure.

1 Introduction

While mainstream acceptance of Bitcoin as an asset appears to be increasing, advanced tools and methods for secure custody of bitcoins are slow to develop. Bitcoin custody encompasses the protection of assets through software, hardware, and operational processes. The foundation of Bitcoin custody is key-management, a well understood topic in the academic literature and in practice. However, Bitcoin custody, in particular multi-stakeholder custody, involves human processes, communication protocols, network monitoring and response systems, software, hardware and physical security environments. Given a secure cryptographic layer, there are still vulnerabilities introduced at the application layer by software developers, at the hardware layer throughout the supply chain, and at the operations layer by users. Without adequate risk management frameworks for custodial operations, Bitcoin users are likely to suffer unexpected losses whether they self-custody funds or employ a third-party custodian.

Open-source custody protocols are emerging [25,5,38,39] and are a critical ecosystem component for improving security standards. If a custody protocol stands to public scrutiny and offers a high-level of security without relying on proprietary processes, users, insurance companies and regulators can have more confidence in it. The emerging custody protocols are trying to reconcile the needs of traditional businesses and banking with Bitcoin's novel identity-less and irreversible transaction properties. A lack of available and accepted open-source custody protocols means that organisations are heavily relying on third-party custodians, or deploying their own custody protocol.

We propose an attack modelling technique as the basis for a risk framework for Bitcoin custody operations, using the Revault protocol[3] as a case-study [4,25]. While the process of model construction is intensive, the resultant framework is extensible and modular and some of its components can be re-used with different custody protocols. It is intended to be readily comprehensible, and, given sufficient validation, the framework can be used by any organisation intending to deploy Revault to better understand their risk posture.

Risk quantification frameworks address several ecosystem problems. Organisations that control bitcoins or other digital assets need accurate models to engage in realistic risk-management. The complexity of custodial risks leaves insurance companies guessing rather than systematically estimating when pricing their insurance offerings or assessing particular solutions for digital custody. Finally, emerging regulatory standards for custody [9,26] are simple and fail to capture advanced custody architectures or enable context-specific risk analyses that acknowledge the full security environment of a custody operation.

The remainder of this paper is structured as follows. Section 2 summarises the components and processes of the Revault protocol. Section 3 discusses our evaluation criteria for an operational risk framework, and introduces the attack-tree formalism on which our risk model is based. Section 4 presents our operational risk model for Revault. Section 5 concludes this paper.

2 Overview of Revault Custody Protocol

Revault is a multi-party custody protocol that distinguishes between *stakeholders* and fund *managers*. The primary protection for funds is a high-threshold multi-signature Script controlled by the stakeholders. The day-to-day operational overhead of fund management is simplified by enabling portions of funds to be delegated to fund managers. Stakeholders define spending policies in-line with traditional controls of expenses, and have automated servers to enforce their policies. In addition, a deterrent is withheld by each stakeholder to mitigate incentives to physically threaten the stakeholders. To achieve this, Revault makes use of sets of pre-signed transactions coupled with an active defence mechanism for detecting and responding to attempted theft transactions. In the following, we will outline the components of the Revault architecture, the transaction set, the stakeholders' routine signing process and the managers' spend process. Refer to [4] for the detailed specification of the open-source protocol.

2.1 Revault Architecture Components

Each stakeholder and manager has a *hardware module* (HM) to manage their private keys and generate signatures for transactions. A backup of private keys is stored for each HM in a separate protected physical environment.

[3] Specifically, the version identified as 609b40dda07155abe5cd4a5af77fc2211d11fbc1 which can be found on the open-source repository hosted on Github [4].

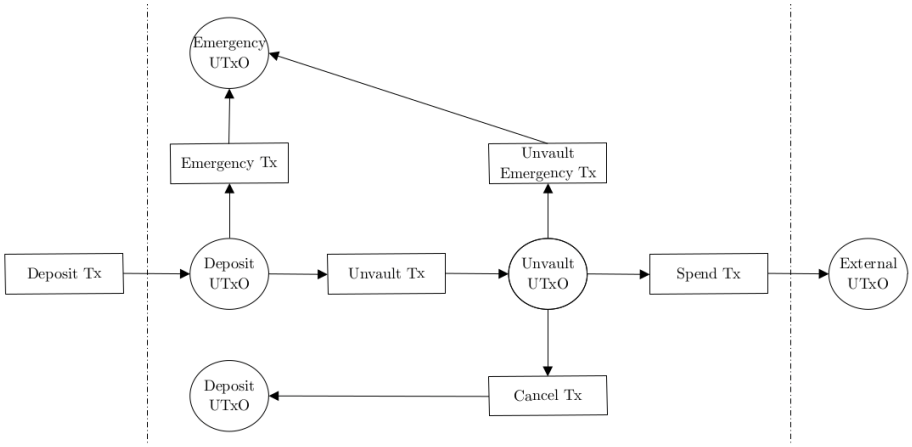

Fig. 1. Diagram of the transaction (Tx) set structure in the Revault protocol. An Unspent-transaction-output (UTxO) is created by a preceding Tx and is consumed by an input in a proceeding Tx.

Each stakeholder and manager uses a *wallet* software to track their co-owned bitcoins, craft transactions, store transaction signatures and communicate with each other through a *coordinator*. The coordinator is a proxy server that simplifies communication for the multi-party wallet. All communication uses Noise KK encrypted and authenticated channels [33].

Stakeholders each have one or more *watchtower*, an online server that enforces the stakeholder's spending policy limitations. Stakeholders each have an *anti-replay oracle* server.

2.2 Revault Transaction Set

The use of hierarchical deterministic wallets means that each participant in the Revault protocol has a tree of public and private keys [41]. To discuss ownership of bitcoins, we refer to a generalisation of a locking Script, called a *descriptor*. The wallet will have multiple addresses that correspond to a single abstracted descriptor. Funds are deposited into the multi-party wallet through a Deposit transaction (Tx) output that pays to the deposit descriptor, describing $N-$signatories locking Scripts derived from the stakeholders (*stk*) extended public keys (*xpub*). In descriptors language formalisation [2] it is defined as:

```
thresh(N, stk_1_xpub, stk_2_xpub, ..., stk_N_xpub)
```

The set of transactions prepared with stakeholders' wallets and signed using their hardware modules (HM) include the Emergency Tx, Unvault Tx, Unvault-Emergency Tx and Cancel Tx. The managers can only prepare and sign a Spend Tx type. Figure 1 depicts these transactions and the essential unspent-transaction-outputs (UTxOs) they create or consume.

An Emergency Tx locks funds to an emergency descriptor which is unspecified by the Revault protocol and is kept private among stakeholders. The descriptor must however be harder to unlock than the deposit descriptor. This is the deterrent for physical threats to the stakeholders.

An Unvault Tx consumes the deposit UTxO and creates an unvault UTxO locked to the unvault descriptor,

```
or( thresh(N, stk_1_xpub, stk_2_xpub, ..., stk_N_xpub),
    and( thresh(K, man_1_xpub, ..., man_M_xpub),
         and( thresh(N, oracle_1_xpub, ..., oracle_N_xpub),
              older(X) ) ) ),
```

that is redeemable by either the N stakeholders *or* the M managers (*man*) along with N automated anti-replay oracles after X blocks.

A Cancel Tx consumes the unvault UTxO and creates a new deposit UTxO. The watchtowers' role is to broadcast the Cancel Tx if a fraudulent spend attempt is detected (either through an unauthorised attempt at broadcasting an Unvault Tx or if a Spend Tx does not abide by the spending policy). The timelock gives watchtowers X blocks worth of time to broadcast a Cancel Tx. An Unvault-Emergency Tx consumes the unvault UTxO and locks funds to the emergency descriptor. It has the same purpose as the Emergency Tx, only it consumes the unvault UTxO rather than the deposit UTxO. A Spend Tx is used by managers to pay to external addresses.

2.3 Stakeholders' Signing Routine

Stakeholders' wallets routinely check for new deposits and each one triggers a signing routine. Figure 2 shows the connections and message types for an example Revault deployment enacting the signing routine. The wallet crafts an Emergency Tx and requests the stakeholder to sign it using their HM. The stakeholder will verify the emergency descriptor on the HM before authorising the signature generation[4]. The wallet then connects to the coordinator to push its signature and will fetch other stakeholders' signatures.

Optionally, stakeholders may also sign the Cancel, Unvault-Emergency, and Unvault Txs to securely delegate funds to the managers. In this case the signing process is the same but is carried out in two steps: first, the signatures for the Cancel and Unvault-Emergency Txs are exchanged with the other stakeholders through the coordinator and then shared with the watchtower(s), and only then are the Unvault Tx signatures shared with managers.

2.4 Managers' Spending Process

Most spending policies cannot be inferred from the Unvault Tx alone and so the Spend Tx must be known to the watchtower to validate an unvaulting attempt. In these cases the Spend Tx must be advertised to the watchtowers before

[4] This feature is not available with current HMs, but integrating compatibility with descriptors (along with other security features) would improve the human-verification component of HM security and is being discussed on the *bitcoin-dev* mailing list [24].

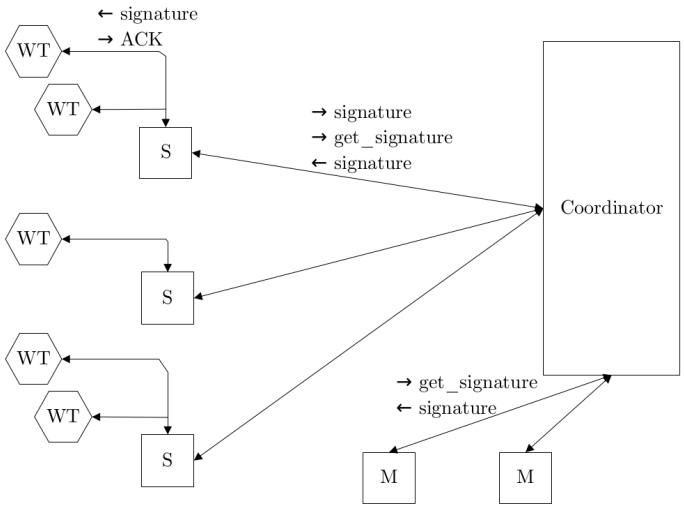

Fig. 2. Data flow diagram for the communication of the *stakeholders' signing routine* with an example Revault deployment. There are three stakeholders (S) who each have one or two watchtowers (WT). There are two managers (M) and a coordinator. Signature messages, signature requests and watchtower acknowledgements (ACK) are only shown once per connection type but apply to each connection of that type (e.g. there is {← signature, → ACK} between each WT and S).

unvaulting, otherwise it will be cancelled. The anti-replay oracle is required to avoid the Spend Tx being modified by the managers *after* the unvault time-lock expires and thus by-passing enforcement of the watchtowers' spending policies.

Any manager can initiate a spend. Figure 3 depicts the spend process . The initiator creates a Spend Tx, verifies and signs it using their HM and passes it back to the wallet in the partially-signed Bitcoin Tx (PSBT) format [10]. It's exchanged with a sufficient threshold of the other managers to add their signatures and hand it back to the initiator. The initiator requests a signature from each of the anti-replay oracles and pushes the fully-signed Spend Tx to the coordinator. The initiator broadcasts the Unvault Tx, triggering a lookup from the watchtowers to the coordinator for the Spend Tx. If the Spend Tx is valid according to *all* of the watchtowers policies and none of them cancel this unvaulting attempt, the manager waits X blocks and broadcasts the Spend Tx.

If, during the unvaulting process, there's a significant increase in the fee-level required for a Spend Tx to be mined, a manager needs to bump the fee. Managers use a dedicated single-party fee wallet for this. Similarly, watchtowers use a fee wallet in the case there is high demand for block space to bump the fee for Cancel or emergency Txs.

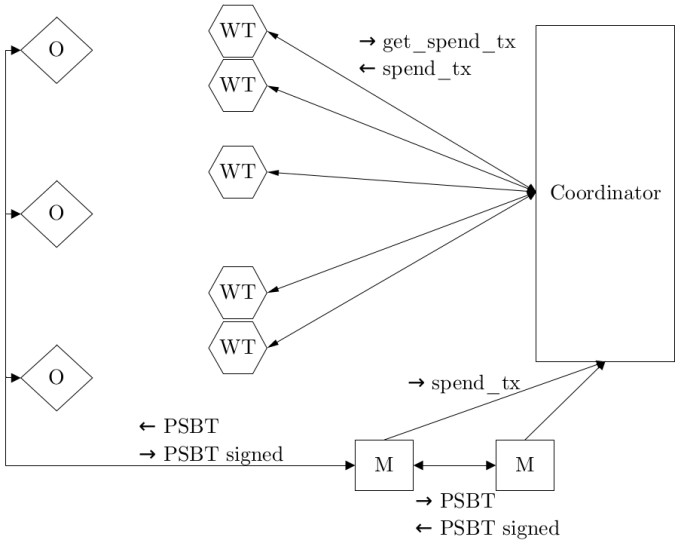

Fig. 3. Data flow diagram for the communication of the managers' spend process. In this example there are two managers (M), three anti-replay oracles (O), five watchtowers (WT) and a coordinator. A Partially-signed Bitcoin Tx (PSBT) is exchanged among managers and between a manager and the anti-replay oracles. A fully signed SpendTx is shared with the WTs through the coordinator.

3 Methodology

To see where this research fits in to the big picture, consider the key life-cycle of a custodial operation. There are three phases; initialization, operation, and termination. Initialization is where wallet and communication keys are generated, where software integrity is verified, hardware security modules are checked, and relevant public information is shared among participants. Operation encompasses the active fund management. Termination is the phase wherein the wallet is de-commissioned and all sensitive information destroyed. Initialization and termination are out of scope for this paper. Our risk model covers the operations phase. In the following we present our rationale for our chosen attack modeling formalism and explain how this can be used as a risk framework.

3.1 Operational Security Models

A framework for high-level risk analyses for the integration of custody into a multi-stakeholder context has not yet been presented. To-date the literature has focused primarily on cryptographic security modeling, dealing with low-level risks associated with cryptographic primitives, key-management protocols, HMs and single-party wallets. The underlying cryptographic security is fundamental but should be complemented by an operational security model, which is much

more likely to be the domain where participants create vulnerabilities for an attacker. Advanced custody protocols that use multi-layer access control with both static and active defences for insider and external attackers demand a whole-system approach to security analysis.

We present now several requirements for our modelling formalism: a) the ability to represent complex processes with numerous components and sequential events; b) supports qualitative risk analysis; c) supports automated quantitative methods for multi-attribute risk analysis; d) readily comprehensible and visual models that are more amenable to open-source intelligence; and e) extensible and modular models to support differential analysis and re-use of modules.

The two most popular attack modeling techniques in cyber-security literature are attack-trees and attack graphs [23]. In short, tools for attack graphs tend to produce graphs that aren't readily comprehensible due to the complexity of real-world attack scenarios [14]. That is, attack graphs don't scale well [35]. On the other hand, attack-trees seem to meet all of our requirements, at least when considered with the right structure and semantics (as described in section 3.2) and thus we construct our risk model using this formalism.

While a statement such as 'our custody solution is based on an $m-$of$-n$ security model' can entail a lot for simple multi-signature custody protocols, it doesn't capture the reality nearly as well as our proposed methodology would. It is certainly not sufficient for a more complex custody protocol like Revault. What is the physical environment for those n private keys? Are any of those keys online? Are there key backups and, if so, what protections are in place for these? Too much depends on the broader security environment of a custody protocol for it to be left without scrutiny.

Application threat modelling has been used to harden the Revault protocol throughout both its theoretical development and implementation. For each application process (spend, routine signing, emergency, revault) a component-by-component and connection-by-connection analysis has been carried out to determine the consequences of outages, data tampering, component corruption, etc., and has resulted in the design specification [4] and the transaction flow threat model [25]. The application threat modeling approach is complementary and has informed us in enumerating the risks presented with the attack-trees. However, in contrast to attack-trees, it lacks a semantic structure which is amenable for automated risk quantification and thus isn't suitable as the basis of a risk framework.

3.2 Attack-Tree Formalism

The risk model is presented using the formalism of attack-trees [36,40,6]. Attack-trees have an attack at their root, and branches that capture alternate (OR) and complementary (AND) attack pathways comprised of intermediate attack goals as non-leaf nodes and basic attack steps as leaf nodes. As in numerous other works [11,18,22,28,27,29], we extend the basic attack-tree to support sequential conjunction of branches (SAND) allowing us to model an attack where some sub-tree of an attack pathway has to occur before and in addition to another

sub-tree. For brevity we depict our attack-trees as nested lists. The logical gates (OR, AND, SAND) shown with each node apply to the next node at the same depth. This means that at any given depth, a node with a SAND gate occurs *before* other nodes that are shown below it. Some aspects of the system are built to be resilient to attack and failure through redundancy. For example, an attacker needs to compromise all stakeholders' private keys to steal funds locked to the deposit descriptor. To be concise, rather than having several copies of the same sub-tree we write (X times) to note that the sub-tree has to happen X times. During an analysis, these sub-trees should be considered as X separate AND sub-trees, since they are contextually different (corresponding to different participants, remote and physical environments).

We provide a set of attack-trees, capturing prominent risks that have been enumerated primarily by considering tangible and intangible assets. Tangible assets (bitcoins) are distinguished by the access control structures determined by the set of descriptors. We consider operational privacy and business continuity as intangible assets.

Our work here is focused on security, rather than safety. In principle, the same methodology could be extended to an integrated security-safety model by constructing attack-fault-trees [22]. Another common extension to the attack-tree formalism is to include countermeasures, producing attack-defence-trees [19,16]. The benefit of our modular modelling technique is that it enables future work to integrate these extensions and re-use results from this work. Hence, we prioritise constructing a strong foundation based only on attacks, and aim to incrementally improve on the model presented as new intelligence emerges.

3.3 On Risk Analysis

Our purpose in constructing the risk model presented in section 4 is to provide a framework to support both qualitative and quantitative risk analyses for specific deployment instances of Revault custody. By determining costs, likelihoods, and other attributes for risks associated with custodial processes, an organisation can perform a differential analysis of countermeasures until their risk-tolerance is satisfied. An explicit framework not only helps an organisation deploying Revault with risk-management but could form a standard by which insurance companies and regulators consider specific deployments. As with any model of complex reality, attack-trees are imperfect and cannot capture every possible attack pathway, but the alternative—complete ignorance—is not better.

To perform a context-specific risk analysis, a set of estimates are made (using in-house empirical data, public research, and expert opinion) for each basic attack step on different attributes such as monetary cost, execution time, or likelihood. With that, a bottom-up procedure (from leaf nodes to the root) is used to compute aggregated attributes. Bayesian methods can be used to update prior estimates with more refined values as new data sources emerge. The process for generating estimates is critical and should be considered with care. In-depth research-based practical guidance on this topic is given by D. W. Hubbard and R. Seiersen in [17]. Given specific contextual information, estimations can be

improved by further decomposing basic attack steps (e.g. 'steal keys backup') into multiple steps (e.g. 'bribe manager to determine backup location' SAND 'break into safe'). If a basic attack step has a highly uncertain estimate, then further decomposition into more explicit steps can be beneficial. On the other hand, decomposing into quantities that are more speculative than the first could compound uncertainty rather than reducing it.

Various methods for analysis can be used to compute aggregated attributes for attack-trees. Kordy *et al.* gave an overview [20]. Our purpose here is to provide the framework on which to perform analyses rather than to provide a specific analysis. We have not performed a comprehensive evaluation of analysis methods, but offer some suggestions based on a comparison in [21]. Two methods that support evaluating the attributes of cost, probability, and time are stochastic-model checking [22] and game-theoretic analysis [16]. Whichever methods are used must appropriately capture the constraints of our model (including SAND gates) and should be automated to enable rapid attribute-based queries for security metrics such as; the expected attack pay-off for the most likely attack, or the possible attack pathways given a budget of $10,000.

Our approach to constructing the risk model is centered on assets since these are clearly distinguished through Bitcoin descriptors, as continuity of a custodial process, or as operational privacy. However, when performing the risk analysis it can be insightful to consider attacker personas [37]: a crime syndicate; an opportunistic burglar; a nation state; a business competitor; or even an insider. If the organisation understands any of these personas well (arguably they should especially understand their competitors and employees) they can reduce the uncertainty in their aggregate risk estimates for these scenarios. Attacker-profiles are a useful way to prune attack-trees [21].

4 Risk Model

We have constructed the risk model with several assumptions that limit the scope of the analysis to the operational aspects of custody. Known risks from other protocol and environment dependencies that are discussed in other works should be considered as complementary but are, for the purpose of clarity, assumed to be benign here. First, we assume that the Bitcoin network is functional, realising its live-ness and availability properties [12,13,32,8,7]. We assume that there is significant hash-rate to prevent blockchain reorganisations of a depth higher than the Unvault Tx's relative lock-time. Next, we assume that Revault's Tx model is robust; with scripts that realise the access control structures we expect, without unintended consequences from Tx malleability and network propagation issues as described in [25]. We assume the initialization process was secure and safe; private keys and backups were correctly and confidentially constructed for each participant, software and hardware integrity were verified, relevant public key information for both the wallet and communication was shared among participants leading to a correct configuration for the wallet clients, watchtowers, anti-replay oracles and the coordinator. We assume that Revault's communica-

tion security model as described in [4] is robust. That is, where messages need to be authenticated or confidential, they are. We assume that the software development life-cycle of Revault is secure, such that any deployment is using an implementation that adheres to the protocol specification. Finally, we assume that entities constructing Deposit Txs don't succumb to a man-in-the-middle attack. That is, they lock funds to the deposit descriptor rather than to an attacker's address.

4.1 Common Attack Sub-Trees

These attack sub-trees are common to different attacks on Revault, and **a**, **b**, **c**, **d**, **e**, **f** and **g** are likely to be common to attacks on other custody protocols.

a : **Compromise a participant (stakeholder or manager)**
- 1 : Coerce participant (OR)
- 2 : Corrupt participant

Coercion and insider threats from corrupt participants must be considered. Legal defences for malicious employee behaviour can be effective deterrents here.

b : **Compromise a participant's (stakeholder's or manager's) HM**
- 1 : Physical attack of HM (OR)
  - *1.1* : Determine location of participant's HM (SAND)
  - *1.2* : Access the physical security environment of the participant's HM (SAND)
  - *1.3* : Exfiltrate keys (either on premise or after stealing it) (OR)
  - *1.4* : By-pass PIN and make the HM sign a malicious chosen message
- 2 : Remote attack of HM (OR)
  - *2.1* : Compromise a device that is then connected to the HM (SAND)
    - *2.1.1* : (see **g**) Compromise the participant's wallet software (OR)
    - *2.1.2* : Trick participant into connecting their HM to a compromised device via social engineering
  - *2.2* : Exploit a firmware vulnerability (OR)
  - *2.3* : Trick participant into compromising their own HM with the user interface of the compromised device
- 3 : (see **a**) Compromise a participant

c : **Compromise a participant's (stakeholder's or manager's) keys backup**
- 1 : Physical Attack (OR)
  - *1.1* : Determine location of the keys backup (SAND)
    - *1.1.1* : Watch the participant between the custody initialization and the start of operations (OR)
    - *1.1.2* : Watch the participant during a backup check (OR)
  - *1.2* : Access the physical security environment of the keys backup (SAND)
  - *1.3* : Depending on backup format, steal or copy it
- 2 : (see **a**) Compromise a participant

d : **Compromise a server (watchtower, anti-replay oracle, or coordinator)**
- 1 : Remote attack (OR)
  - *1.1* : Exploit a software vulnerability (OR)

1.1.1 : Determine the public interfaces of the server (SAND)
1.1.2 : Exploit a vulnerability on one of the softwares listening on these interfaces
1.2 : Exploit a human vulnerability (e.g. trick participant into performing a malicious update)
2 : Physical attack (OR)
2.1 : Determine server's location (SAND)
2.2 : Access the physical security environment of the server (SAND)
3 : (see **a**) Compromise the participant managing the server

An attacker who successfully completes **d** for a watchtower will be able to steal funds from the watchtower's fee wallet and will be able to force an emergency scenario by broadcasting all Emergency and Unvault-Emergency Txs it has stored. They can also prevent broadcast of a Cancel Tx from this watchtower either passively (*ACK* the secure storage of the signature to the stakeholder, but then drop the signature) or actively.

e : Shutdown a watchtower
1 : Physical attack on the watchtower (OR)
1.1 : Determine watchtower's location (SAND)
1.2 : Sever the internet connection to the building in which the watchtower is located (OR)
1.3 : Sever the power-line connection to the building in which the watchtower is located (OR)
1.4 : Access the physical security of the watchtower and un-plug the machine
2 : Remote attack on the watchtower
2.1 : Determine public interfaces of watchtower (SAND)
2.2 : Denial of Service attack through one of the public interfaces (OR)
2.3 : Eclipse attack on the watchtower's Bitcoin node [15] (OR)
2.3.1 : Slowly force de-synchronisation of watchtower with the true block height by delaying block propagation [34] (OR)
2.3.2 : Prevent outgoing propagation of Cancel or Emergency Txs
2.4 : Denial of Service attack on the fee-bumping UTxOs pool—not enough funds to pay competitive fees (OR)

f : Get signature from participant to unlock UTxO for Theft Tx
1 : (see **a**) Compromise a participant (OR)
2 : (see **b**) Compromise participant's HM (OR)
3 : (see **c**) Compromise participant's keys backup

g : Compromise a participant's wallet
1 : Physical attack (OR)
1.1 : Locate participant's device (SAND)
1.2 : Access physical security environment of participant's device
2 : Remote attack (OR)
2.1 : Determine public interfaces of device (SAND)
2.2 : Exploit a vulnerability
3 : (see **a**) Compromise participant

Participant's wallet devices are expected to be used for day-to-day activities. With many vulnerabilities to exploit, the likelihood of success for **g** is high.

h : **Determine the locking Script for a deposit or unvault UTxO (*Witness Script*)**
- 1 : (see **g**) Compromise any participant's wallet
- 2 : (see **d**) Compromise a watchtower
- 3 : (see **d**) Compromise an anti-replay oracle

Deposit and unvault descriptors are deterministic, but public keys are needed to derive UTxO locking Scripts. These are stored by all wallets, watchtowers and anti-replay oracles.

i : **Satisfy an input in a Theft Tx that consumes an identified deposit UTxO or unvault UTxO (through N−of−N)**
- 1 : (see **h**) Determine the UTxO locking Script (*Witness Script*) (SAND)
- 2 : Prevent the relevant Emergency Tx from being broadcast until the Theft Tx is confirmed (where $A + B = N$) (AND)
  - *2.1* : (see **d**) Compromise a watchtower (A times)
  - *2.2* : (see **e**) Shutdown a watchtower (B times)
  - *2.3* : (see **g**) Compromise stakeholder's wallet (N times)
- 3 : (see **f**) Get signature from a stakeholder to unlock UTxO for Theft Tx (N times)

j : **Satisfy an input in a Theft Tx that consumes an identified unvault UTxO (through K−of−M, anti-replay oracles and time-lock)**
- 1 : (see **h**) Determine the UTxO locking Script (*Witness Script*) (SAND)
- 2 : Receive signatures for Theft Tx from all N anti-replay oracles (AND)
  - *2.1* : Compromise a manager's private communication keys and the set of anti-replay oracles' public communication keys (OR)
    - *2.1.1* : (see **g**) Compromise a manager's wallet (OR)
    - *2.1.2* : (see **a**) Compromise a manager
  - *2.2* : (see **d**) Compromise the anti-replay oracle
- 3 : (see **f**) Get signature from a manager to unlock UTxO for Theft Tx (K times)

k : **Satisfy an input in a Theft Tx that consumes an identified emergency UTxO**
- 1 : Determine the emergency descriptor policy (SAND)
- 2 : Satisfy the emergency descriptor's locking conditions (may include waiting for time-locks, giving sufficient signatures, giving hash pre-images, *etc.*)

The details of the emergency descriptor are intentionally not specified with the Revault protocol, except that it is more difficult to access than the deposit descriptor. Stakeholders may compartmentalise and distribute the descriptor information to afford its privacy some resilience to attack.

4.2 Attack-Trees

The following attack-trees are the foundation for an operational risk framework for Revault.

A : **Compromise privacy of the custody operation (determine the set of public UTxOs)**

1 : (see **d**) Compromise any of the servers (OR)
2 : (see **a**) Compromise a participant (OR)
3 : (see **g**) Compromise a participant's wallet (OR)
4 : Traffic analysis of connections between servers and/or wallets (OR)
5 : Blockchain analysis

Without privacy support for advanced descriptors (such as by using MuSig2 [30] or MuSig-DN [31] if the proposed Taproot [1] upgrade is activated by the Bitcoin network) Revault's operational privacy is brittle.

B : Broadcast Theft Tx(s) that consume all deposit UTxOs
1 : (see **A**) Determine $\mathcal{D}$, the set of deposit UTxOs (SAND)
2 : (see **h**) Determine the locking Script for deposit UTxO ($|\mathcal{D}|$ times)
3 : (see **i**) Satisfy an input in a Theft Tx that consumes an identified deposit UTxO ($|\mathcal{D}|$ times)

A Theft Tx that consumes all available deposit UTxOs would be catastrophic since this comprises the majority of funds. We recommend a defence wherein each stakeholder is equipped with a panic button that is directly connected to their watchtower or dedicated emergency service. When triggered, all the signed Emergency and Unvault-Emergency Txs are broadcast, negating the pay-off for an attacker and thus acting as a deterrent.

C : Broadcast Theft Tx(s) that consume as many available unvault UTxOs as watchtower spending policies permit
1 : Determine spending constraints of all watchtowers' policies (SAND)
1.1 : (see **a**) Compromise a participant (OR)
1.2 : (see **g**) Compromise a manager's wallet
1.3 : (see **d**) Compromise a watchtower (N times)
2 : Determine $\mathcal{U}$, the set of available unvault UTxOs (SAND)
2.1 : (see **A**) Compromise privacy of the custody operation (determine the set of public UTxOs) (SAND)
2.2 : (see **h**) Determine the locking Script for unvault UTxO ($|\mathcal{U}|$ times)
3 : (see **i** OR **j**) Satisfy an input in a Theft Tx that consumes an identified unvault UTxO ($|\mathcal{U}|$ times)

C can be avoided if watchtowers have a white-list of addresses that Spend Txs can pay to.

D : Broadcast Theft Tx(s) that consume all available unvault UTxOs, bypassing watchtowers' spending policies
1 : Prevent watchtower from broadcasting Cancel or Unvault-Emergency Txs before Theft Tx is confirmed (N times SAND)
1.1 : (see **d**) Compromise a watchtower (OR)
1.2 : (see **e**) Shutdown a watchtower
2 : Determine $\mathcal{U}$, the set of available unvault UTxOs (SAND)
2.1 : (see **A**) Compromise privacy of the custody operation (determine the set of public UTxOs) (SAND)
2.2 : (see **h**) Determine the locking Script for unvault UTxO ($|\mathcal{U}|$ times)
3 : (see **i** OR **j**) Satisfy an input in a Theft Tx that consumes an identified unvault UTxO ($|\mathcal{U}|$ times)

E : **Broadcast a Theft Tx that by-passes watchtowers' spending policies**
- 1 : Determine $\mathcal{U}$, the set of available unvault UTxOs (SAND)
  - *1.1* : (see **A**) Compromise privacy of the custody operation (determine the set of public UTxOs) (SAND)
  - *1.2* : (see **h**) Determine the locking Script for unvault UTxO ($|\mathcal{U}|$ times)
- 2 : (see **f**) Get signature from a manager to unlock $U \subseteq \mathcal{U}$, a subset of available unvault UTxOs for a valid Spend Tx (K times)
- 3 : (see **i** OR **j**) Satisfy an input in a Theft Tx that consumes an identified unvault UTxO ($|U|$ times)
- 4 : (see **d**) Compromise an anti-replay oracle to get a signature for the valid Spend Tx which consumes U, the UTxOs (N times SAND)
- 5 : Advertise the valid Spend Tx to the watchtowers through the coordinator (SAND)
- 6 : Broadcast all Unvault Txs that the valid Spend Tx depends on and wait for the time-lock to expire

F : **Force emergency scenario**
- 1 : Broadcast the full set of signed Emergency and Unvault-emergency transactions
  - *1.1* : (see **d**) Compromise a watchtower (OR)
  - *1.2* : (see **a**) Compromise a stakeholder

The emergency deterrent results in better security from the most egregious physical threats to participants (particularly stakeholders who control the majority of funds) but also in a fragility to the continuity of operations that could be abused by an attacker. Attacks that rely on **E** may seek a pay-off other than fund theft, such as damaging the reputation of the organisation for having down-time and taking a leveraged bet on the likely market consequences. However, forced down-time attacks through power or internet outages or detainment of personnel are prevalent threats for organisations who aren't deploying Revault. In any case, with this risk model the consequence of not using an emergency deterrent can be considered by performing an analysis with pruned attack-trees.

G : **Broadcast a Theft Tx which consumes all available UTxOs locked to the emergency descriptor**
- 1 : (see **F**) Force an emergency scenario (SAND)
- 2 : Determine $\mathcal{E}$, the set of available emergency UTxOs (SAND)
  - *2.1* : (see **A**) Compromise privacy of the custody operation (determine the set of public UTxOs)
- 3 : (see **k**) Satisfy an input in a Theft Tx that consumes an identified emergency UTxO ($|\mathcal{E}|$ times)

H : **Broadcast a Theft Tx which spends from a manager's fee wallet**
- 1 : (see **g**) Compromise a manager's wallet

While this is a relatively simple attack, the fee wallet will never hold a significant portion of bitcoins and is considered external to the custody protocol.

I : **Prevent Emergency, Unvault-Emergency, and Cancel Tx valid signature exchange**
- 1 : 1 of N stakeholders doesn't sign (OR)

1.1 : Prevent stakeholder from accessing their HM (OR)
1.2 : Prevent stakeholder from accessing their wallet (OR)
1.3 : (see **a**) Compromise a stakeholder
2 : Shutdown coordinator (OR)
3 : (see **e**) Shutdown a watchtower (N times) (OR)
4 : Blockchain re-organization and Deposit Tx outpoint malleation.

J : **Prevent Unvault Tx signature exchange**
1 : 1 of N stakeholders doesn't sign (OR)
1.1 : Prevent stakeholder from accessing their HM (OR)
1.2 : Prevent stakeholder from accessing their wallet software (OR)
1.3 : (see **a**) Compromise a stakeholder
2 : Shutdown coordinator (OR)
3 : Prevent all managers from accessing their wallet software

K : **Prevent managers from broadcasting a Spend Tx**
1 : Prevent managers from signing the Spend transaction (OR)
1.1 : (see **d**) Compromise an anti-replay oracle (OR)
1.2 : Prevent sufficient threshold of managers from signing the Spend Tx (where $A + B + C = M - K + 1$) (OR)
1.2.1 : (see **a**) Compromise a manager (A times)
1.2.2 : Prevent manager from accessing their HM (B times)
1.2.3 : Prevent manager from accessing their wallet software (C times)
2 : Force broadcast of Cancel Tx (OR)
2.1 : (see **d**) Compromise a watchtower
3 : Prevent broadcast of Unvault Tx
3.1 : High demand for block space making the Unvault Tx not profitable to mine.[5]
3.2 : (see **g**) Compromise manager's wallet (M times)

5 Conclusion

The rise of Bitcoin has led to a new commercial ecosystem, with market exchanges enabling its sale and purchase, companies and financial institutions offering secure custody services, and insurance brokers and underwriters willing to insure individuals, exchanges and custodians against loss or theft of their assets . In this paper we first posit that a methodology to better understand risks in custodial operations is needed, something complementary to understanding blockchain and cryptographic security. We put forth requirements of the modelling technique and propose attack-trees as a formalism which satisfies those requirements. We exemplify the approach by presenting a library of attack-trees constructed for a multi-party custody protocol called Revault and explain how this framework can be used as a basis for risk-management in custodial operations. The next steps for this work are to: construct a set of defences to the prominent risks and incorporate them into the model; and to determine or build a suitable tool for automating computations for a specific analysis.

[5] Manager's fee-bumping wallet can not cover this until a network policy such as Package Relay [3] is implemented.

Acknowledgements

We thank Professor McBurney (King's College London), Kevin Laoec (Wizard-Sardine) for insightful conversations and for reviewing the text.

Funding

Funding is gratefully acknowledged under a UK EPSRC-funded GTA Award through King's College London and from WizardSardine.

References

1. (Bitcoin Improvement Proposal) Taproot: SegWit version 1 spending rules, `https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki`, last accessed 29 Jan 2021
2. Output Script Descriptors: a language for abstracting out the spending conditions of a Bitcoin transaction output, `https://github.com/bitcoin/bitcoin/blob/master/doc/descriptors.md`, last accessed 26 Jan 2021
3. Package Relay design questions for the Bitcoin P2P network, `https://github.com/bitcoin/bitcoin/issues/14895`, last accessed 29 Jan 2021
4. Practical Revault: A specification for the initialization and operation of the Revault custody protocol, `https://github.com/re-vault/practical-revault`
5. Glacier design document (2017), `https://glacierprotocol.org/assets/design-doc-v0.9-beta.pdf`, last accessed 10 Jan 2021
6. Amoroso, E.G.: Fundamentals of Computer Security Technology. Prentice-Hall, Inc., USA (1994)
7. Badertscher, C., Garay, J., Maurer, U., Tschudi, D., Zikas, V.: But Why Does It Work? A Rational Protocol Design Treatment of Bitcoin. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 34–65. Springer International Publishing, Cham (2018)
8. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: A composable treatment, vol. 10401 LNCS (2017). https://doi.org/10.1007/978-3-319-63688-$7_1$1
9. Capital Markets and Technology Association: Digital Assets Custody Standard (2020), `https://www.cmta.ch/content/272/cmta-digital-assets-custody-standard-v1-public-consultation.pdf`, last accessed 10 Jan 2021
10. Chow, A.: Partially signed bitcoin transaction format (2017), `https://github.com/bitcoin/bips/blob/master/bip-0174.mediawiki`, last accessed 18 May 2020
11. Gadyatskaya, O., Jhawar, R., Kordy, P., Lounis, K., Mauw, S., Trujillo-Rasua, R.: Attack Trees for Practical Security Assessment: Ranking of Attack Scenarios with ADTool 2.0. vol. 9826, pp. 159–162 (08 2016). https://doi.org/10.1007/978-3-319-43425-$4_1$0
12. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin backbone protocol: Analysis and applications. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9057**, 281–310 (2015). https://doi.org/10.1007/978-3-662-46803-6 10

13. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 291–323. Springer International Publishing, Cham (2017)
14. Haque, M.S.: An evolutionary approach of attack graphs and attack trees: A survey of attack modeling (09 2017)
15. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 129–144. USENIX Association, Washington, D.C. (Aug 2015), https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman
16. Hermanns, H., Krämer, J., Krcál, J., Stoelinga, M.: The value of attack-defence diagrams. vol. 9635, pp. 163–185 (04 2016). https://doi.org/10.1007/978-3-662-49635-0_9
17. Hubbard, D.W., Seiersen, R.: How to Measure Anything in Cybersecurity Risk (2016)
18. Jhawar, R., Kordy, B., Mauw, S., Radomirovic, S., Trujillo-Rasua, R.: Attack Trees with Sequential Conjunction. CoRR **abs/1503.02261** (2015), http://arxiv.org/abs/1503.02261
19. Kordy, B., Mauw, S., Radomirovic, S., Schweitzer, P.: Foundations of attack–defense trees. vol. 6561, pp. 80–95 (09 2010). https://doi.org/10.1007/978-3-642-19751-2_6
20. Kordy, B., Pièvre-Cambacédès, L., Schweitzer, P.: Dag-based attack and defense modeling: Don't miss the forest for the attack trees. Computer Science Review **13** (03 2013). https://doi.org/10.1016/j.cosrev.2014.07.001
21. Kumar, R.: Truth or Dare: Quantitative security risk analysis using attack trees. Ph.D. thesis (10 2018). https://doi.org/10.3990/1.9789036546256
22. Kumar, R., Stoelinga, M.: Quantitative Security and Safety Analysis with Attack-Fault Trees (01 2017). https://doi.org/10.1109/HASE.2017.12
23. Lallie, H., Debattista, K., Bal, J.: A review of attack graph and attack tree visual syntax in cyber security. Computer Science Review **35**, 100219 (02 2020). https://doi.org/10.1016/j.cosrev.2019.100219
24. Loaec, K.: Hardware wallets and "advanced" Bitcoin features (2021), https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-January/018352.html, last accessed 19 Jan 2021
25. Loaec, K., Poinsot, A.: Revault: A multi-party Bicoin vault architecture (2020), https://github.com/re-vault/practical-revault/blob/master/revault.pdf
26. M. Sato, M. Shimaoka, H.N.: General Security Considerations for Cryptoassets Custodians (2019), https://tools.ietf.org/html/draft-vcgtf-crypto-assets-security-considerations-05
27. Maynard, P., Mclaughlin, K., Sezer, S.: Modelling Duqu 2.0 Malware using Attack Trees with Sequential Conjunction. pp. 465–472 (01 2016). https://doi.org/10.5220/0005745704650472
28. Maynard, P., McLaughlin, K., Sezer, S.: Decomposition and sequential-AND analysis of known cyber-attacks on critical infrastructure control systems. Journal of Cybersecurity **6**(1) (12 2020). https://doi.org/10.1093/cybsec/tyaa020, https://doi.org/10.1093/cybsec/tyaa020, tyaa020
29. Nguyen, H.N., Bryans, J., Shaikh, S.: Attack Defense Trees with Sequential Conjunction. IEEE (2019)
30. Nick, J., Ruffing, T., Seurin, Y.: Musig2: Simple two-round schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261 (2020), https://eprint.iacr.org/2020/1261

31. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057 (2020), https://eprint.iacr.org/2020/1057
32. Pass, R., Seeman, L., Shelat, A.: Analysis of the Blockchain Protocol in Asynchronous Networks. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 643–673. Springer International Publishing, Cham (2017)
33. Perrin, T.: The Noise Protocol Framework (2018), https://noiseprotocol.org/noise.pdf, last accessed 19 Jan 2021
34. Riard, A., Naumenko, G.: Time-dilation attacks on the lightning network (2020)
35. Schmitz, C., Sekulla, A., Pape, S.: Asset-Centric Analysis and Visualisation of Attack Trees, pp. 45–64 (11 2020). https://doi.org/10.1007/978-3-030-62230-5$_3$
36. Schneier, B.: Attack Trees (1999), https://www.schneier.com/academic/archives/1999/12/attack_trees.html, last accessed 12 Jan 2021
37. Shostack, A.: Threat Modeling: Designing for Security (2014)
38. Square: Subzero (2020), https://subzero.readthedocs.io/en/master/, last accessed 19 Jan 2020
39. Swambo, J., Hommel, S., McElrath, B., Bishop, B.: Custody protocols using bitcoin vaults (2020), https://arxiv.org/abs/2005.11776, last accessed 10 Jan 2021
40. Weiss, J.D.: A system security engineering process. In: Proceedings of the 14th National Computer Security Conf. (1991)
41. Wuille, P.: Hierarchical deterministic wallets (2012), https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, last accessed 18 May 2020