SABRE: Protecting Bitcoin against Routing Attacks

Maria Apostolaki
ETH Zurich
apmaria@ethz.ch

Gian Marti
ETH Zurich
gimarti@student.ethz.ch

Jan Müller
ETH Zurich
jan.m.muller@me.com

Laurent Vanbever
ETH Zurich
lvanbever@ethz.ch

***Abstract*—Nowadays Internet routing attacks remain practically effective as existing countermeasures either fail to provide protection guarantees or are not easily deployable. Blockchain systems are particularly vulnerable to such attacks as they rely on Internet-wide communications to reach consensus. In particular, Bitcoin—the most widely-used cryptocurrency—can be split in half by any AS-level adversary using BGP hijacking.**

In this paper, we present SABRE, a secure and scalable Bitcoin relay network which relays blocks worldwide through a set of connections that are resilient to routing attacks. SABRE runs alongside the existing peer-to-peer network and is easily deployable. As a critical system, SABRE design is highly resilient and can efficiently handle high bandwidth loads, including Denial of Service attacks.

We built SABRE around two key technical insights. First, we leverage fundamental properties of inter-domain routing (BGP) policies to host relay nodes: *(i)* in networks that are inherently protected against routing attacks; and *(ii)* on paths that are economically-preferred by the majority of Bitcoin clients. These properties are generic and can be used to protect other Blockchain-based systems. Second, we leverage the fact that relaying blocks is communication-heavy, not computation-heavy. This enables us to offload most of the relay operations to programmable network hardware (using the P4 programming language). Thanks to this hardware/software co-design, SABRE nodes operate seamlessly under high load while mitigating the effects of malicious clients.

We present a complete implementation of SABRE together with an extensive evaluation. Our results demonstrate that SABRE is effective at securing Bitcoin against routing attacks, even with deployments of as few as 6 nodes.

I. Introduction

Cryptocurrencies, and Bitcoin in particular, are vulnerable to routing attacks in which network-level attackers (i.e., malicious Autonomous Systems or ASes) manipulate routing (BGP) advertisements to divert their connections. Once on-path, the AS-level attacker can disrupt the consensus algorithm by partitioning the peer-to-peer network. Recent studies [17] have shown that these attacks are practical and disruptive. Specifically, *any* AS-level attacker can isolate ∼50% of the Bitcoin mining power by hijacking less than 100 prefixes [17]. Such attacks can lead to significant revenue loss for miners and enable exploits such as double spending.

Network and Distributed Systems Security (NDSS) Symposium 2019
24-27 February 2019, San Diego, CA, USA
ISBN 1-891562-55-X
https://dx.doi.org/10.14722/ndss.2019.23252
www.ndss-symposium.org

Problem Protecting against such partitioning attacks is challenging. On the one hand, local (and easily deployable) countermeasures [17] fail to provide strong protection guarantees. These countermeasures include having Bitcoin clients monitor their connections (e.g., for increased or abnormal delays) or having them select their peers based on routing information. On the other hand, Internet-wide countermeasures are extremely hard to deploy. For example, systematically hosting Bitcoin clients in /24 prefixes (to prevent more-specific prefix attacks) requires the unlikely cooperation of all Internet Service Providers hosting Bitcoin clients in addition to a considerable increase to the size of the Internet routing tables. Worse yet, even heavy protocol modification such as encrypting *all* Bitcoin traffic would not be enough to guarantee Bitcoin safety as AS-level attackers would still be able to distinguish (and drop) Bitcoin traffic using transport headers.

SABRE: A Secure Relay Network for Bitcoin In this paper, we present SABRE, a secure relay network which runs alongside the existing Bitcoin network and which can protect the vast majority of the Bitcoin clients against routing attacks. Unlike existing countermeasures, SABRE secures Bitcoin against routing attacks in a way which: *(i)* provides strong security guarantees to any connected client by enabling it to learn and propagate blocks; *(ii)* is partially deployable; and *(iii)* provides security benefits early-on in the deployment, with as little as two relay nodes. We built SABRE based on two key insights.

Insight #1: Hosting relays in inherently safe locations Our first insight is to host SABRE relay nodes in locations that: *(i)* prevent attackers from diverting relay-to-relay connections, so as to secure SABRE internal connectivity; and *(ii)* are attractive (from a routing viewpoint) to many Bitcoin clients, so as to protect client connections to the relay network. We do so by leveraging a fundamental characteristic of BGP policies, namely, that connections established between two ASes which directly peer with each other and which have no customers cannot be diverted by routing attacks. In SABRE, only such ASes are considered for relay locations.

Using real routing data, we show that such safe locations are plentiful in the current Internet with 2000 ASes being eligible. These ASes include large cloud providers, content delivery networks, and Internet eXchange Points which already provide hosting services today and therefore have an incentive to host SABRE nodes. We also show that SABRE deployments with 6 nodes are already enough to protect 80% of the clients from 96% of the AS-level adversaries (assuming worst case scenario for SABRE).

Insight #2: Resiliency through software/hardware co-design As a publicly-facing and transparent network, SABRE is an obvious target for attackers who could, among others, craft (D)DoS attacks against its publicly-known nodes to disrupt it. To protect SABRE deployments against such attacks, our second insight is to leverage the fact that most of the relay operations are communication-heavy (propagating information around) as opposed to being computation-heavy. In addition to that, the content (block) that the relays need to propagate each time is predictable and small in size. These properties enable us to offload most SABRE operations to hardware, using programmable network devices. Thanks to this software-hardware co-design, SABRE relay nodes can sustain up to Tbps of load.

We show that our relay node design is practical by implementing it in P4 and connecting it to an extended regular Bitcoin client. P4 [19], [15] is a new programming language which allows to specify the behavior of network data planes. Besides being general enough to support SABRE, our analysis indicates that SABRE memory requirements are within the capabilities of today's P4 switches.

Contributions Our main contributions are:

- The design of SABRE, a novel relay network that prevents AS-level adversaries from partitioning it (Section III).
- An algorithm for positioning SABRE nodes in selected ASes so as to minimize chances of a successful routing attack against the Bitcoin system (Section IV).
- A novel software-hardware co-design for SABRE relay nodes which enables them to operate seamlessly under high load (Section V).
- A measurement study showing the effectiveness of SABRE in protecting Bitcoin clients together with the inability of existing relays networks to provide such protection (Section VI).
- A complete implementation of SABRE, including the P4 code to run on programmable network switches [7] along with an extended Bitcoin client that can connect to them (Section VII).
- An analysis of the incentives for candidate ASes to host SABRE nodes (Section VIII).

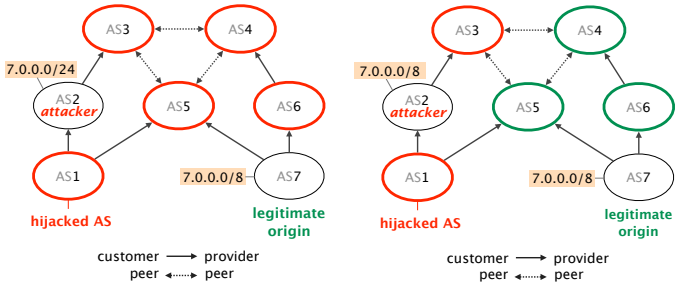
Although SABRE focuses on Bitcoin, its design principles can be applied to protect other Blockchain systems from routing attacks. We discuss the broader applicability of SABRE in Section IX.

II. Background

In this section, we present an overview of BGP and how it can be misused (Section II-A), before introducing Bitcoin and the concept of relays (Section II-B).

A. Border Gateway Protocol (BGP)

The Internet consists of over 60k individual networks, known as Autonomous Systems (ASes), which rely on BGP [48] to exchange information about how to reach 700k+ IP prefixes [10]. Each AS originates one or more IP prefixes which are then propagated AS-by-AS.



(a) AS 2 hijacks `7/24` (b) AS 2 hijacks `7/8`

Fig. 1: The effectiveness of a malicious AS in diverting traffic using BGP hijacks depends on its position and on whether she originates existing prefixes or longer ones. Here, AS 2 attracts traffic from all ASes when originating `7/24` (a), but only from AS 1 and 3 when originating `7/8` (b).

Policies BGP is a single-path and policy-based protocol. Each AS selects one single best route to reach any IP prefix—including self-owned ones—that it selectively exports to its neighboring ASes (omitting the AS from which it learned the route). These selection and exportation processes are governed by the business relationships each AS maintains with its neighbors. The most common business relationships are known as *customer-provider* and *peer-peer* [27]. In a customer-provider relationship, the customer AS pays the provider AS to get full Internet connectivity. The provider provides such connectivity by: *(i)* exporting to the customer all its best routes; and *(ii)* exporting the prefixes advertised by the customer to all its neighbors. In a peer-peer relationship, the two ASes connect only to transfer traffic between their respective customers and internal users. They therefore only announce their own prefixes and the routes learned from their customers to each other. Regarding route selection, an AS prefers customer-learned routes over peer-learned ones and peer-learned routes over provider-learned ones. If multiple equally-preferred routes exist (e.g., if two customers announce a route to the same prefix), an AS favors the route with the minimum AS path length towards the prefix before relying on some arbitrary tie-break [48].

Hijack BGP routers do not validate route advertisements. Any malicious AS can create fake advertisements, known as *BGP hijacks*, for any prefix, and advertise them to its neighbors. Hijacks constitute an effective way for an AS to redirect traffic directed to given destinations.

We distinguish two types of hijacks according to whether the fake announcement contains: *(i)* a more-specific (longer) prefix than a legitimate one; or *(ii)* an existing (equally specific) prefix. In the former case, the hijacker AS will attract *all* the traffic addressed to the more-specific prefix, independently from its position in the Internet topology. This is because routers forward traffic according to the most-specific prefix matching it. In the latter case, the rogue advertisement competes with the legitimate one. The amount of diverted traffic then depends on the relative positions of the attacker and the victim in the Internet topology.

Fig. 1a illustrates an example of a more-specific attack in which AS 2 advertises 7/24, a more-specific prefix of 7/8 which is advertised by AS 7. In doing so, AS 2 effectively redirects the corresponding traffic from *all* ASes except AS 7.[1] In contrast, Fig. 1b illustrates the effect of AS 2 advertising 7/8 alongside AS 7. AS 2 only manages to attract the traffic from AS 1 and AS 3. Indeed, AS 1 learns two routes to 7/8 from its two providers (AS 2 and 5) and prefers the illegitimate one from AS 2 because it is shorter. Similarly, AS 3 prefers to reach 7/8 using the customer route learned via AS 2 over the legitimate peer route learned via AS 5.

More-specific hijacks are more powerful but come with drawbacks. First, such attacks are more visible since the hijacked prefixes propagate Internet-wide. In contrast, existing prefixes propagate in smaller regions [30]. For instance, in Fig. 1b, while AS 4 and AS 5 learn about the hijacked prefix, they do not propagate it further as they prefer the legitimate announcement. Second, network operators often filter BGP advertisements whose prefix lengths are longer than /24 [36]. This filtering effectively prevents more-specific attacks against existing /24s.

By default, hijacking a prefix creates a black hole at the attacker's location. However, the attacker can turn a hijack into an *interception* attack and make herself a man-in-the-middle (MITM) by preserving at least one path to the legitimate origin [47], [31]. For instance, in Fig. 1a, AS 2 could selectively announce 7/8 to AS 1 so as to keep a working path to the legitimate origin via AS 3. Observe that AS 2 cannot achieve the opposite interception attack, i.e., diverting the traffic from AS 3 and redirecting it to AS 1, as it does not learn a path to the legitimate origin via AS 1.

B. Bitcoin

Bitcoin is a decentralized transaction system which relies on a randomized peer-to-peer network to implement a replicated ledger, the *Blockchain*, which keeps track of the ownership of funds and the balance of each Bitcoin address. The Bitcoin network disseminates two types of information: *transactions* and *blocks*. Transactions are used to transfer value from one address to another, while blocks are used to synchronize the state of the system. Bitcoin nodes are identified by their IP address, connect to each other using TCP, and exchange data in plain text. Bitcoin comprises around 10k publicly reachable nodes [9] while 10× more nodes are behind NAT [18].

Blocks are created by *miners* and contain the latest transactions as well as a Proof-of-Work (PoW). A PoW is a computationally-heavy puzzle, unique for every new block, whose difficulty is regularly adapted such that it takes 10 minutes on average to generate a new block [44]. A newly mined block is propagated network-wide and is appended to the blockchain according to consensus, thereby yielding a financial reward to its miner. Bitcoin participants unaware of the latest blocks will waste their mining power and can be fooled into accepting invalid transactions.

[1]Traffic from AS7 itself is not redirected as AS7 relies on internal routing protocols, such as OSPF, to reach its own prefixes.

Relay networks are overlay networks maintained by a single administrative entity which run alongside Bitcoin's peer-to-peer network. Relay networks aim at assisting the Bitcoin network, not replacing it. The three most well-known relays are: Falcon [3], FIBRE [2], and the Fast Relay Network (FRN) [5]. These relay networks aim at speeding up block propagation by relying on a system of high-speed relay nodes and/or on advanced routing techniques. By connecting to these relays, a Bitcoin client can alleviate the effects of bad network performance that may otherwise affect the time needed to acquire a new block.

III. SABRE: A Secure Relay Network for Bitcoin

SABRE is a transparent relay network protecting Bitcoin clients from routing attacks by providing them with an extra secure channel for learning and propagating the latest mined block. By transparent, we mean that the IP addresses of the SABRE relay nodes will be publicly known (e.g. via a website) and that every Bitcoin client is welcome to connect to them. To benefit from SABRE, a Bitcoin client simply needs to successfully establish a connection to at least one relay node. SABRE relays contribute to the block propagation by receiving, validating and transmitting new blocks to all connected clients.

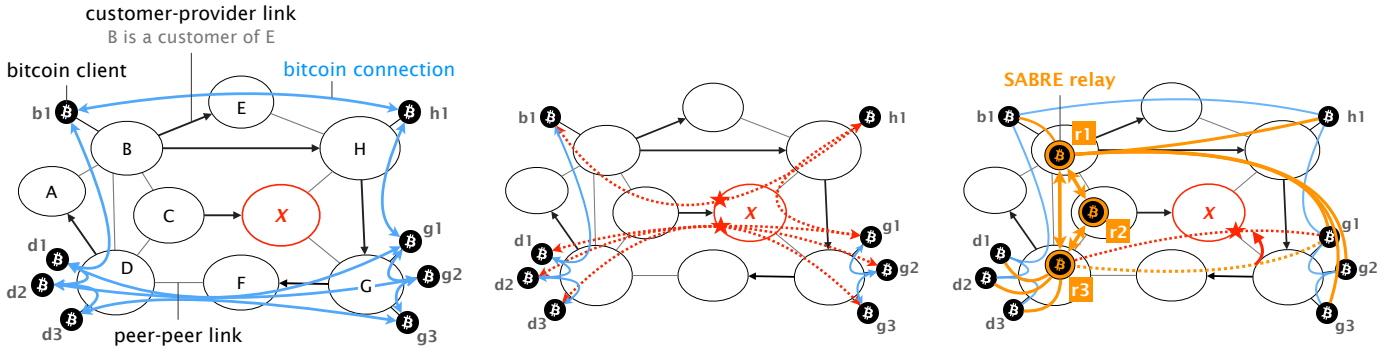
To achieve its goals, the SABRE network *must* remain connected at all times, even under arbitrary BGP advertisements or extremely high load (Section III-A). SABRE leverages two key insights to guarantee ceaseless operation: *(i)* smart positioning of the relay nodes to secure its internal connections and minimize the clients attack surface (Section III-B); and *(ii)* a hardware/software co-design to enable relay nodes to sustain almost arbitrary load (Section III-C).

A. Attacker Model

We consider a single AS-level attacker whose goal is to partition the Bitcoin network into two disjoint components S and N. To do so, she first diverts the traffic destined to S or N by performing an interception attack using existing and more-specific prefixes (Section II). The attacker then: *(i)* identifies the Bitcoin connections by inspecting the network and/or transport layer headers (i.e., by matching on IP addresses and/or TCP/UDP ports); and *(ii)* drops the connections bridging the partition. Such an attack is powerful and can effectively partition the Bitcoin network [17] causing revenue losses and allowing double spendings. In fact, partitioning is an effective attack against any Blockchain system as it prevents nodes from communicating (see Section IX).

We assume that the attacker knows: *(i)* the IP addresses of *all* SABRE nodes; along with *(ii)* the code running on them. As such, the attacker can hijack the prefixes hosting relay nodes and drop *all* traffic destined to them. Alternatively, the attacker can issue multiple requests to the relay nodes effectively performing a (D)DoS attack.

Example We illustrate our attacker model using Figure 2a and 2b which depict a simple AS-level topology composed of 9 ASes. ASB, ASD ASH and ASG host Bitcoin clients which establish Bitcoin connections between each other (in blue). ASX is malicious and aims at disconnecting the nodes on the left side ($S = \{b1, d1, d2, d3\}$) from the others ($N = \{h1, g1, g2, g3\}$). To that end, ASX intercepts the



(a) AS-level topology

(b) AS X hijacks ASH & ASG

(c) With SABRE, network stays connected

Fig. 2: SABRE protects the Bitcoin network from AS-level adversaries aiming to partition it. Without SABRE, AS X can split the network in half by first diverting traffic destined to AS H and AS G using a BGP hijack and then dropping the corresponding connections (Fig. 2b). With SABRE, the network stays connected (Fig. 2c).

Bitcoin connections directed to N by hijacking ASH and ASG prefixes. As a result, ASX diverts all the connections from S to N, and some more (e.g., the connection from $h1$ to $g1$). We depict the diverted connections in red in Figure 2b. Once on-path, the attacker drops the Bitcoin traffic *crossing* the partition and forwards the rest normally. For instance, the attacker does *not* drop the connection between $h1$ and $g1$ and simply relays it from ASH to ASG untouched. Once the attack is launched, nodes in S can no longer communicate with nodes in N: the Bitcoin network is partitioned.

B. SABRE secure network design

We now explain the routing properties behind SABRE relay locations and how they protect relay-to-relay, client-to-relay, and relay-to-client connections from being disconnected by routing attacks. We describe an algorithm for systematically finding such locations in Section IV.

Protecting relay-to-relay connections A SABRE deployment is composed of relays hosted in /24 prefixes which belong to ASes that: *(i)* have no customer; *(ii)* peer directly; and *(iii)* form a k-connected graph. These constraints protect relay-to-relay connections from routing attacks for three reasons.

First, these constraints prevent any attacker from diverting traffic between relays by advertising a more-specific prefix, effectively forcing her to advertise existing prefixes and thus compete with legitimate advertisements. *Second*, these constraints prevent any attacker from diverting relay traffic away from the ASes hosting relay nodes by advertising an economically-preferred route. Indeed, the ASes hosting relays follow the advertisements of their direct peers to reach each other and have no customers (i.e. no better advertiser AS). As such, the number of malicious ASes which can advertise an equally-preferred route are limited to those that directly peer with the ASes hosting relays. Finally, these constraints make the chances for effective attackers to divert relay connections to exponentially decrease as the connectivity of the relay graph k increases. Indeed, BGP routers rely on an arbitrary tie-break to select among equally-preferred routes (e.g., by choosing routes learned from the lowest peer address [48]). Assuming that the attacker is equally likely to win this tie-breaking, she would only have a 3.1% (0.5^5) probability of disconnecting a 5-connected relay network. In Section VI, we show that well-connected relay networks are numerous.

Protecting client-to-relay connections While we can host relays in cherrypicked ASes, we cannot choose which ASes host Bitcoin clients. Concretely this means that AS-level adversaries hijacking relay prefixes can still divert connections originated by Bitcoin clients *to* the relays.

In SABRE, we protect client-to-relay connections by further restricting the locations in which we host relays to ASes whose /24 advertisements tend to be preferred by ASes with Bitcoin clients. Doing so we can lower the amount of traffic malicious ASes can divert, i.e. maximize SABRE's coverage. While individual relays locations are unlikely to protect many Bitcoin clients against all possible attackers, we show that a relatively small set of relays often can (Section VI). This design decision is motivated by the observation that Bitcoin clients are concentrated in few hosting ASes [17].

Protecting relay-to-client connections Finally, an attacker might try to divert the traffic sourced from the relay network *to* the Bitcoin clients. For instance, an attacker could hijack the prefixes of Bitcoin clients and drop the relay connections by matching on any relay IP address. While this technique is more cumbersome for the attacker (there are way more clients than relays), it is nonetheless possible. SABRE prevents this attack by obfuscating the traffic exchanged between the clients and the relay nodes. This obfuscation forces the attacker to perform full inspection (beyond L4 headers) on a possibly huge volume of diverted traffic, rendering the attack highly impractical. Observe that while encrypting the already-obfuscated traffic would render even full inspection useless, encryption alone would not help as the attacker would still be able to distinguish Bitcoin traffic by matching on the destination IP.

SABRE relies on two techniques to obfuscate relay traffic. First, the relays can modify their source IP addresses when communicating with Bitcoin clients. This is possible as SABRE uses connectionless communications between the relays and the clients, enabling clients to accept packets with a different source IP than the one they send traffic to. Second, Bitcoin

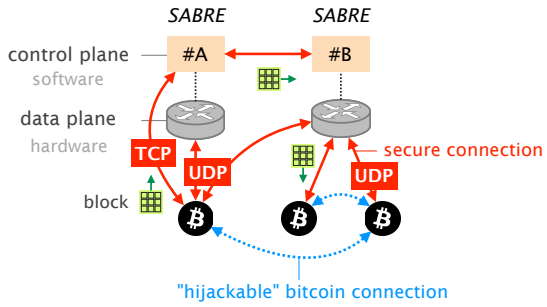


Fig. 3: SABRE offloads most communication to the switch

clients can connect to SABRE relays via a VPN/proxy service. Doing so would force the attacker to first find the mapping between the proxy IP and the corresponding Bitcoin client.

Example Using Fig. 2c, we now explain how a SABRE deployment of three relays, namely $r1$, $r2$ and $r3$, protects against routing attacks such as the one shown in Fig. 2b by securing intra-relay connectivity and maximizing coverage.

With respect to Fig. 2a, each Bitcoin client is now connected to at least one relay node in addition to maintaining regular Bitcoin connections. Here, nodes g_1, g_2, g_3 are connected to relay r_1 while node g_1 is also connected to node r_3. Hosted in ASes that peer directly, relay-nodes protect their internal connectivity against ASX's hijacks. For instance, consider that ASX advertises the $/24$ prefix covering $r1$ to ASC. Since ASX is a provider of ASC, ASC discards the advertisement as it prefers to route traffic via a peer instead. At the same time, forming a 2-connected graph allows the relay network to sustain any single link cut. A link cut can be caused by a failure, an agreement violation or an unfiltered malicious advertisement from another direct peer.Observe that this would not hold if r_2 was not deployed. Finally, the exact positioning of relays is such that the paths towards them are more preferred over those of the attacker. As an illustration, ASX can divert the connection from ASG to ASD by advertising a more attractive path to ASG (as a peer) than the one it originally uses (a provider route, via ASF). Even so, ASX cannot divert the connection from ASG to ASB. Indeed, ASG will always prefer its customer path over any peer path.

C. SABRE resilient software/hardware node co-design

As a transparent and accessible relay network, SABRE nodes should be able to sustain high load, either caused by legitimate Bitcoin clients or by malicious ones who try to exhaust their resources. To scale, SABRE nodes rely on a software/hardware co-design in which most of the operations are offloaded to programmable network switches (e.g., P4-enabled ones). We illustrate this deployment in Fig. 3 where two SABRE nodes are connected to each other and to some Bitcoin clients. One client talks directly to the controller via the switch while the others talk only to the switch. Observe that a software-based implementation of the relay node would also protect Bitcoin against routing attack, yet it will be more prone to (D)DoS attacks since it will have 2–3 orders of magnitude lower throughput [34] compared to a hardware-based one.[2]

[2]We discuss the possibility of a software deployment of SABRE in §VIII.

SABRE's relay design is based on the observations that: *(i)* the content that needs to be cached in the relay node is predictable and small in size, consisting in the one or two blocks of 1MB that were most recently mined; and *(ii)* most of the relay operations are communication-heavy, consisting in propagating the latest known block to many clients and distinguishing the new ones. The former allows for effective caching while the latter allows for a partially hardware implementation in programmable network devices. This software/hardware co-design enables SABRE nodes to operate at Tbps and therefore sustain large (D)DoS attacks. Indeed, Barefoot Tofino programmable network devices can deal with as much as 6.5 Tbps of traffic in the backplane [7].

While using programmable network devices enable high performance, it does not make it easy due to the lack of a broad instruction set and the strict limitations with respect to memory and number of operations per packet. We overcome these limitations with three techniques. First, our software/hardware design seamlessly blends in hardware and software operations, allowing to automatically escalate operations that cannot be done in the switch to a software component. In SABRE, only the validation of new blocks (which happens once every 10 minutes) needs to be escalated while all other requests are served by the hardware over a UDP-based protocol. Second, our implementation relies on optimized data structures which are both memory efficient and require a fixed number of operations per access. Third, we heavily precompute and cache values that would need to otherwise be computed on the switch (e.g., UDP checksums).

IV. SABRE Secure Network Design

In this section, we first formally define the problem of selecting the ASes to host SABRE relays in (Section IV-A) so as to minimize the probability of a successful routing attack against Bitcoin. We then describe an algorithm for solving this problem (Sections IV-B and IV-C).

A. Problem Statement & Challenges

The security provided by SABRE depends on: *(i)* how secure the intra-relay connectivity is, i.e., how many connections an AS-level adversary needs to hijack in order to disconnect the graph; and *(ii)* how much of the Bitcoin network is covered, i.e., how likely it is that an AS-level adversary will be able to prevent clients from connecting to *all* relay nodes.

We take both factors into consideration while constructing a SABRE network by first setting the desirable level of intra-relay connectivity (e.g., 2-connectivity), and then finding the relay ASes that will maximize the Bitcoin coverage. Formally, we define our problem as follows:

Problem statement Let $G = (\mathcal{AS}, E)$ be the AS-level topology graph in which vertices ($\mathcal{AS}$) correspond to ASes and edges (E) to inter-AS links. Let also $\mathcal{B} \subseteq \mathcal{AS}$ be the subset of ASes that host Bitcoin clients and $\mathcal{R} \subseteq \mathcal{AS}$ be the subset of ASes that have no customers. Finally, let $G' = (\mathcal{R}, E')$ be the subgraph of G induced by $\mathcal{R}$ that contains the subset $E' \subseteq (\mathcal{R} \times \mathcal{R})$ of peer-to-peer inter-AS links. We define $\mathcal{A} = \mathcal{AS} \times \mathcal{B}$ as the set of all attack scenarios, namely all pairs of ASes (x, v) in which AS x acts as AS-level adversary

for AS v which hosts Bitcoin clients. Let $\mathcal{S} : \mathcal{R} \rightarrow \mathcal{A}$ be a function which, given a candidate relay, finds the subset $\alpha \subseteq \mathcal{A}$ of attack scenarios that this candidate relay protects against. Let furthermore $\mathcal{C} : \mathcal{P}(\mathcal{A}) \rightarrow \mathbb{R}$ be a function ($\mathcal{P}(\cdot)$ denotes the power set) which, given a set of attack scenarios $\alpha \subseteq \mathcal{A}$, quantifies their significance for the Bitcoin system by computing the number of Bitcoin clients hosted in victim ASes, i.e. $\mathcal{C}(\alpha) = \sum_{(x,v)\in\alpha} w_v$ where w_v is the number of Bitcoin clients affected by the attack scenarios in α.

Given a desired number of relays N and a desired inter-relay connectivity k, our problem is to maximize the number of attack scenarios Bitcoin clients are protected against. Formally, we aim at finding G'', a subgraph of G' induced by $\mathcal{R}'$, such that $\mathcal{R}' \subseteq \mathcal{R}$; $|R'| = N$; G'' is k-connected; and $\mathcal{C}\big(\bigcup_{r_i\in R'} \mathcal{S}(r_i)\big)$ is maximized.

Challenges Solving the above problem optimally is challenging for at least three reasons. First, the amount of clients protected by any subset of relays R' depends on the *union* of the sets of the attack scenarios each relay $r \in R'$ protects against. As these are in general not disjoint, this problem reduces to the maximum coverage problem. Second, finding k-connected subgraphs in a random graph is difficult [22]. Third, in order to find the attack scenarios a relay protect against, one needs to predict the forwarding path each AS with Bitcoin clients will use to reach each canditate-relay considering *any* possible attacker.

We develop a heuristic to address the first two challenges (Section IV-B) and an algorithm for finding the possible attack scenarios for every attacker (Section IV-C).

B. Positioning SABRE Relays

As described above, positioning relays optimally maps to solving a maximum coverage problem. Given the complexity, we rely on a greedy approach which is shown to be effectively optimal for the maximum coverage problem [26].

Our algorithm starts with an empty set R' and the set of candidate ASes $\mathcal{R}$ which satisfy the constraints listed in Section III-B and are also contained in at least one k-connected subgraph of at least N nodes, as only those can host one of the relay nodes of a k-connected network of N relays. It then iteratively adds relays to the set R' aiming at maximize the number of covered attack scenarios while preserving k-connectivity for R'. This simple procedure runs in $\mathcal{O}(N)$ and works well in practice (Section VI).

In particular, in each round, we consider as candidates the set $\mathcal{R}_k \subseteq \mathcal{R} \setminus R'$ which are connected with at least $\min\{k, |R'|\}$ of the already-selected ASes in R'. Then we add the candidate $r \in \mathcal{R}_k$ that adds the maximum weighted coverage to R', i.e., the one with the maximum $\mathcal{C}\big(\bigcup_{r_i\in(R'\cup r)} \mathcal{S}(r_i)\big) - \mathcal{C}\big(\bigcup_{r_i\in R'} \mathcal{S}(r_i)\big)$ and we update R' accordingly, i.e, $R' := R' \cup \{r\}$. When we have selected all candidates, so that $|R'| = N$, we return R'.

We show in Section VI that the resulting relay networks can readily protect between 80% to 98% of the existing Bitcoin clients (depending on the internal connectivity and number of deployed nodes) from 99% of the potential attackers. The exact algorithm for positioning the relay nodes can be found in the Appendix A1.

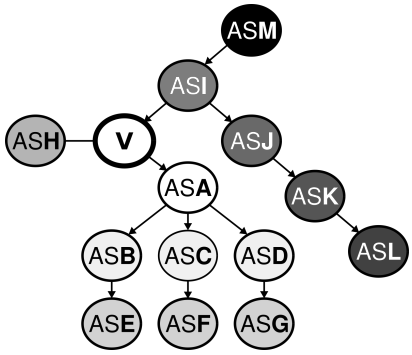


Fig. 4: Shades illustrate ASV routing preference ranging from white (most preferred) to black (least preferred). Traffic from ASV to preferred AS is less likely to be hijacked.

C. Calculating covered attack scenarios

Having explained how we can position SABRE relays based on the attack scenarios they cover, we now describe how we compute these scenarios for each relay, i.e., how we implement the function $\mathcal{S}$. More specifically, we describe how we predict, for each AS hosting Bitcoin clients and each AS-level adversary, whether the hosting AS will prefer the advertisements coming from the attacker AS over legitimate announcements coming from relay ASes.

Our algorithm is based on the observation that, to check whether a attacker AS (say ASM) can divert traffic from a victim AS (say ASV) to to a relay AS (say ASR), we only need to compare the path from ASV to ASR and from ASV to ASM. If the path to ASM is more preferred then ASM can successfully hijack traffic from an ASV to ASR. Path preference is dictated by the business relationships established between ASes together with the path length: customers are preferred over peers, peers over providers, and shorter paths over longer ones.

As an illustration, Figure 4 illustrates an AS topology in which arrows indicate business relationships: providers are drawn above their customer (ASV is the provider of ASA), while peers are drawn alongside each other (ASH and ASV are peers). The different shades indicate which advertisements ASV prefers, ranging from white (most preferred) to black (least preferred). ASV prefers advertisements learned from ASA (its customer) over advertisements learned from ASH (its peer) or from ASI (its provider). Likewise, ASV prefers advertisements from ASE over the ones originated by ASD. While both are learned via ASA (its customer), ASD is closer to ASV (2 hops) than ASE (3 hops). Intuitively, Figure 4 depicts for any two ASes ASX and ASY whether ASV would prefer ASX's advertisements over ASY's, should both advertise the same prefix. For instance, if a relay is hosted in ASH (ASV peer), all the ASes that ASV can reach via customer links (ASA–ASG) are possible attackers. Similarly, if the relay is hosted in ASM then any other ASes (ASA–ASL) can divert the corresponding traffic from ASV.

We describe the algorithm we use to compare the BGP preference of two paths with a common start in Appendix A2.

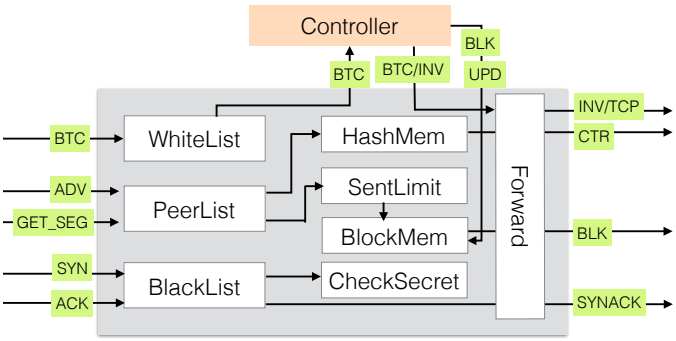
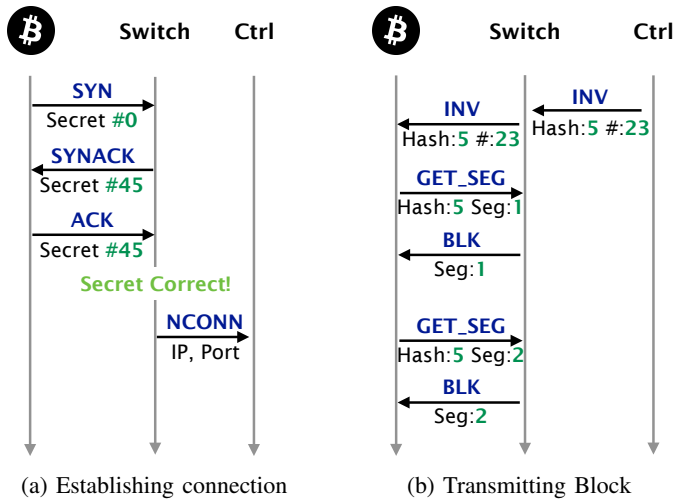


Fig. 5: The switch intercepts all incoming traffic, answers to all UDP requests and redirects TCP traffic of whitelisted clients to the controller. The switch contains the latest mined Block in *BlockMem* and multiple components to track the connected and banned clients (e.g.White/Black List, Connected) and detect attacks (e.g. CheckSecret, SentLimit)



(a) Establishing connection (b) Transmitting Block

Fig. 6: (a) BTC client establishes a connection with the switch using a 3-way handshake. (b) Relay advertises a new block INV via the switch and transmits it using multiple BLK messages after client requests using GET_SEG messages.

V. SABRE Resilient Relay Node Design

We now explain the software/hardware co-design of a SABRE node (Section V-A) and its operations (Section V-B) which ensure that the node's resources are not maliciously exhausted and that benign clients are not denied service.

A. Hardware/Software Co-Design

Figure 5 illustrates SABRE's software/hardware co-design. It is composed of a programmable switch connected to a modified Bitcoin client which acts as a controller.

The switch is responsible for: *(i)* serving client connections; *(ii)* protecting the controller only from malicious clients; *(iii)* propagating blocks; and *(iv)* distinguishing new blocks from old ones. In contrast, the controller is responsible for validating new blocks, advertising them to the connected clients and updating the switch memory accordingly.

Bitcoin clients establish UDP connections with the switch and (rarely) regular Bitcoin connections (over TCP) with the controller. Switches only allow approved Bitcoin clients to establish connections with the controller. As most clients "consume" blocks rather than producing them, we expect most clients to only interact with SABRE's hardware component.

SABRE defines a UDP-based protocol to facilitate communication between the Bitcoin clients and the switch as well as between the switch and the controller. The protocol is composed of 8 messages. Five of them are exchanged between the switch and the clients: SYN, SYN/ACK, ACK, GET_SEQ, and ADV. The three remaining are sent between the switch and the controller: NCONN, UPD, and BLK.

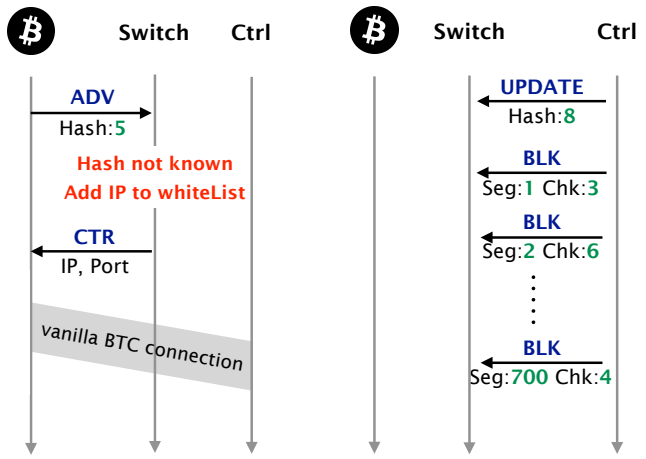
Similarly to TCP, SYN, SYN/ACK, ACK are used to prevent spoofing attacks. GET_SEQ, BLK and ADV relate to block management. Specifically, GET_SEQ enables a client to request a particular segment of a block which is sent as a BLK, while ADV enables a client to advertise a newly mined block to the relay. The switch sends a NCONN to notify the controller of new connections. The controller sends an UPD followed by a BLK message to update the switch with the latest block.

The switch maintains three data structures to manage client connections and track down anomalies: *PeerList*, *Whitelist*, and *Blacklist*. *PeerList* contains information about connected clients, i.e., those who successfully completed the three-way handshake. Similarly, *Whitelist* stores clients that are allowed to communicate with the controller directly. *Blacklist* contains clients that have misused the relay and are banned. The switch also maintains one data structure to store the latest block(s): *BlockMem*. *BlockMem* is composed of indexed equal-sized segments of a block together with precomputed checksums for each segment. This data structure allows the switch to timely reply with the requested segment avoiding additional computations. Moreover, the switch contains two components devoted to anomaly detection: *SentLimit* and *CheckSecret*. *SentLimit* detects clients that requested a block too many times. *CheckSecret* verifies during the handshake that a client is using its true IP. Finally, the switch also maintains one data structure for checking whether an advertised hash is new or known: *Memhash*.

In the following, we describe the different operations performed by the relay and how each of them modifies each of the data structures. In Section VII, we show that our design can sustain 1M malicious and 100k benign client connections with less than 5 MB of memory in the switch. This memory footprint is only a fraction of the memory offered by programmable switches today [35], allowing the switch to be used for other applications.

B. Relay operations

We now describe SABRE relay operations. The client and controller are extended versions of the default Bitcoin client and the switch is implemented in P4 [19]. Our protocol defines four operations: *(i)* how regular Bitcoin clients connect to a relay node; *(ii)* how a relay node propagates blocks back to them; *(iii)* how a relay node receives and validates blocks



(a) Receiving block (b) Updating switch

Fig. 7: (a) BTC client advertises a new block identified as unknown by the switch. The client gets white-listed, allowing it to connect directly to the controller. (b) If the received block is valid, the controller updates the switch using an UPDATE message carrying the block's hash followed by a BLK messages carrying the data.

transmitted by the clients; and *(iv)* how the controller updates the switch memory upon receiving a new valid block.

Managing client connections In order to avoid spoofing attacks, Bitcoin clients initialize connections to relay nodes using a three-way handshake as shown in Figure 6a. As for a normal TCP connection, the client first sends a SYN packet. Upon receiving the SYN, the switch echoes back a secret value calculated using the client's IP address and UDP port in a SYN/ACK packet. The client then includes this secret value in the final (ACK) packet as a proof that it owns the source IP address that it is using.

Upon successfully completing the handshake, the switch adds an entry for the client's IP and port number in the *PeerList* and notifies the controller via a NCONN message. The *PeerList* is implemented as a Bloom filter (BF) for memory efficiency. Doing so, the switch verifies that an incoming packet belongs to an established connection and drops it otherwise. As BFs do not support listing all inserted items, the controller stores the connections for future use (e.g., advertising new blocks and updating the *PeerList*).

Learning new blocks Relay nodes need to learn new blocks that are mined. New blocks are transmitted to the relays from regular clients. Being a network device with limited computational capabilities, the switch is unable to validate blocks. Thus, advertised blocks need to be transmitted to the controller after they have been filtered by the switch.

As illustrated in Fig. 7a, the node advertises a block by its hash to the switch using an ADV message. The switch checks whether the hash is already known using the HashMem. If the hash is not known, then the switch asks the client to connect to the controller with a CTR message and stores its IP in the *Whitelist*. If the transmitted block is legitimate the client's IP will stay in the whitelist for four days. The client connects to the controller as if it was a regular Bitcoin client, while the switch forwards the TCP traffic to the controller. The switch only allows packets from white-listed clients to reach the controller. Observe that a malicious miner cannot monopolize or overload the controller with its connections as even a pool with 30% of the hash power cannot keep more than 172 whitelisted clients at any given moment.[3]

Even so, a malicious miner might still try to engineer block races by flooding the relay node with multiple blocks simultaneously which will need to be validated by the controller. To shield against this attack, the switch keeps the number of active nodes that are white-listed. When this number exceeds a predefined threshold (set based on the controller's hardware capabilities), the switch will stop whitelisting new clients. In this case, the controller receives blocks from the nodes that are already whitelisted. Indeed, these nodes are diverse enough, with respect to mining power origin, to keep the relay up-to-date, thanks to the expiry mechanism in the *Whitelist*. For instance, any pool with at least 0.17% of mining power can keep at least one node in the *Whitelist* forever. In essence, the switch implements a simple-yet-efficient reputation-based access-list to protect the controller from Sybil attacks.

Updating the switch with a new block If a newly-transmitted block is valid, the controller updates the switch's memory with a new mapping of segment IDs to data segment that corresponds to a particular block hash. The switch can then transmit the segments to the clients upon requests. Observe though that the switch sends data to a UDP socket. Thus, the IP and UDP checksums need to be correct for the packet to be accepted. The UDP checksum is calculated using a pseudo-header and the one's complement sum of the payload split into 16 bits segments. Since computing this in the switch would result in repetitive and unnecessary computations, we precompute the one's complement sum of the block segment and cache it together with the segment itself. Using this value the switch needs only to add the header parts that are different per client to calculate the checksum.

Figure 7b illustrates the sequence of packets the controller sends to update the switch. Initially, it sends an UPD message containing the new hash. This first message tells the switch to prepare its state for the new block. The next messages are sent to transmit each of the segments of the block as well as the precomputed one's complement sum.

Propagating a newly-learned block The relay node advertises new blocks to all its connected clients who can then request a block segment-by-segment. Blocks are transmitted in multiple individual segments for three reasons: *(i)* to allow clients to request lost segments independently; *(ii)* to avoid loops in the data plane which would otherwise be necessary as the block does not fit in one packet; and *(iii)* to protect against amplification attacks.

As illustrated in Figure 6b, the controller sends an INV message which is forwarded by the switch. This INV message contains the hash of the new block as well as the number of

[3]Every day, 144 Blocks are mined (on average). For each block at most one node is whitelisted (the one that is not already whitelisted and advertised the Block first)

segments it is composed of. In the example, the relay advertises hash #5 which is composed of 23 segments. If the Bitcoin client is unaware of the advertised block, it requests it using a GET_SEG message containing the hash of the block and each of the 23 segment IDs. In the example, the client first requests the segment of ID:1 of the block with hash #5 then the segment of ID:2 and so on. If either the GET_SEG or the SEG is lost the client will simply request the corresponding segment again. As a protection mechanism, the switch bans clients that request a block multiple times. To that end, all requests traverse a heavy-hitter detector, namely *SentLimit*. For efficiently implementing this component, one can reuse [49] which can operate with just 80KB of memory.

VI. Network Architecture Evaluation

In this section, we evaluate SABRE's efficiency in protecting Bitcoin against routing attacks. Specifically, we answer the following questions: How effective is SABRE in preventing routing attacks targeted against the entire network and individual clients? How does this effectiveness change with the size and the connectivity of the SABRE network? How does SABRE stand out against other relay networks and known counter-measures?

We found that even a small deployment of 6 single-connected SABRE nodes can prevent 94% of ASes in the Internet from isolating more than 10% of the Bitcoin clients; while larger deployments of 30 relays that are 5-connected can prevent more than 99% of the ASes from isolating more than 20% of Bitcoin clients. In addition, we show that existing relay networks, like Falcon [3] and FIBRE [2], offer *no* protection against routing attacks. Finally, we show that SABRE provides security level on-par with hosting all clients in /24, an effective but clearly impractical countermeasure.

We start the section by describing our methodology (Section VI-A) before presenting our results in detail.

A. Methodology

Datasets Our evaluation relies on a joint dataset combining routing and Bitcoin information. Regarding routing information, we rely on the AS-level topology and AS-level policies provided by CAIDA [1], collected in May 2018. We use the routing tree algorithm [31] to compute the forwarding path followed between any two ASes. We assume that the attacker's advertisements are systematically picked at the tie-breaking state of the BGP decision process (the worst-case for SABRE) [4]. Regarding Bitcoin information, we use the IPs of Bitcoin clients from [8] along with the IPs of existing relay nodes from [2], [3], both collected in May 2018. We merge the two datasets by associating each Bitcoin IP to the AS advertising the most-specific IP prefix covering it (using the routes collected by RIPE BGP collectors [4]).

B. SABRE security efficiency

SABRE protects against network-wide partitions To evaluate how effective SABRE is against adversaries that wish to partition the Bitcoin network, we quantify how likely it is for a random adversary to be able to disconnect multiple clients from the relay network. The fraction of clients a particular AS can disconnect from the relays is the maximum set of Bitcoin clients she can isolate, as Bitcoin nodes connected to the relay network cannot be partitioned.

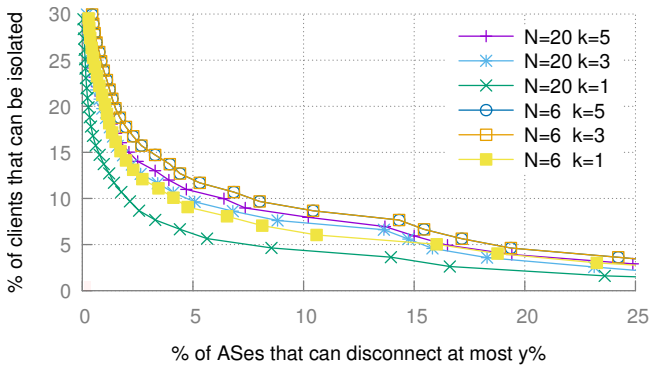


Fig. 8: Less than 2.5% of ASes are able to disconnect more than 15% of clients (N: the number of deployed relays; k: relay-graph connectivity; Tie breaks in favor of the attacker).

Fig. 8 illustrates how protected the Bitcoin network is depending on the size N and internal connectivity k of the SABRE network. The graph shows, for each given fraction y of Bitcoin nodes, the percentage of ASes that would be able to independently disconnect it from SABRE. For $N = 20, k = 1$, less than 3% of ASes are able to prevent a considerable fraction of Bitcoin clients (15%) from connecting to the relay network. In contrast, more than 90% of the clients can be isolated by *any* AS in the current network [17].

The mapping between the number of possible attackers and the partition sizes varies with the size and connectivity of SABRE. In particular, increasing the number of deployed nodes decreases the chances that adversaries can divert traffic successfully. On the other hand, decreasing the intra-connectivity requirements (i.e., the value of k) allows our algorithm (Section IV) to select from a larger set of relays and thus to form a more effective SABRE. This creates an interesting trade-off between how secure the intra-relay connectivity is and how well the relay nodes cover the existing Bitcoin network. For example, while a SABRE of 6 relays that are connected in full-mesh (5-connected graph) is extremely hard to partition, as the AS-level adversary would need to divert 5 peer-to-peer links, it enables more AS-level adversaries to disconnect a larger part of Bitcoin clients from SABRE. For example, 3% of ASes can potentially create a partition including 22% of Bitcoin nodes. In contrast, a 1-connected SABRE allows fewer attackers to perform severe attacks—only 1% of ASes could create a 12% partition—but the relay network itself can be partitioned by a single link failure or a successful hijack from a direct peer.

SABRE protects most individual clients To evaluate how effectively SABRE protects individual clients, we look at how likely it is for Bitcoin clients to be prevented by a random AS-level adversary from reaching *all* relay nodes.

[4]Results for the opposite case, where tie-breaking systematically picks paths originated by relay ASes, can be found in Appendix B

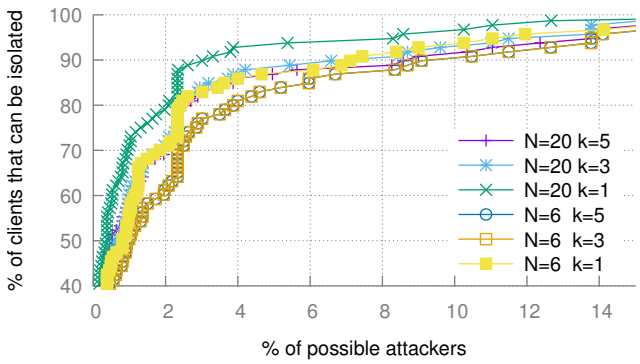


Fig. 9: 85% of the clients are protected against 96% of possible attackers (Tie breaks in favor of the attacker)

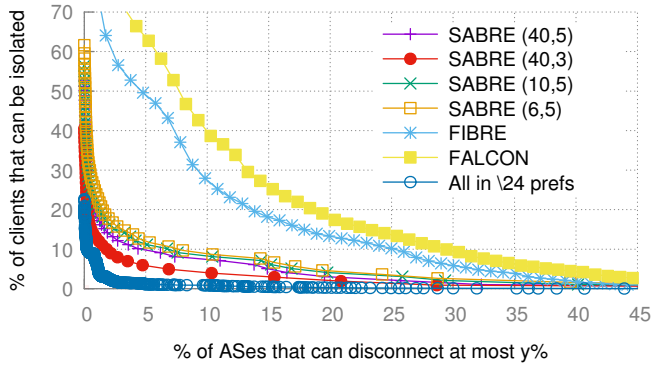


Fig. 10: SABRE is far more secure than deployed relays and very close to the unemployable alternative countermeasure of hosting all clients in /24. (Tie breaks in favor of the attacker)

Fig. 10 shows, for each given percentage of ASes in the Internet, the percentage of Bitcoin clients could be attacked and disconnected from SABRE by this percentage of ASes. We see that 80% of the clients are protected from 96% of the AS-level adversaries even with a SABRE network of only 6 nodes that are 5-connected. There is again a trade-off between secure intra-connectivity and the coverage of Bitcoin clients. For example, a SABRE of 6 1-connected nodes protects 90% of Bitcoin clients from 92.5% of ASes, while a fully connected 6-node SABRE protects from only 89.5% of ASes. Interestingly, increasing connectivity from $k = 3$ to $k = 5$ does not decrease the protected clients significantly while making disconnecting the relay network almost impossible.

C. SABRE efficiency compared to existing relay networks

We compare SABRE to FIBRE [2] and Falcon [3] with respect to their effectiveness against routing attacks. We found that SABRE outperforms both, for three key reasons.

Existing relays are vulnerable to longer-prefix hijacks *All* relay nodes of both FIBRE and Falcon are hosted in prefixes that are shorter than /24. As such, *any* AS-level adversary can cut connections among relays as well as from the Bitcoin clients only by hijacking 6 more-specific prefixes for FIBRE and 10 for Falcon.

Existing relay networks are poorly connected Even if these relay networks were hosted in /24 prefixes, our analysis revealed that their connections could still be diverted by same-prefix advertisements. In particular, we found that FIBRE relays would be disconnected by any of 652 ASes, and Falcon by any of 3 ASes even if /24 prefixes were used.

Existing relays provide bad coverage Even ignoring their poor relay-to-relay connectivity and again assuming that these relay networks were hosted in /24 prefixes, their client-to-relay connections would still have been more vulnerable than those of SABRE allowing for more network-wide and targeted attacks. We compare existing relay networks with SABRE with respect to how well they protect against routing attacks using the same graphs as in § VI-B. In particular, Fig. 10 shows the percentage of ASes that are able to independently isolate a fraction of the Bitcoin network as a function of this fraction while Fig. 11 shows the cumulative percentage of clients as a function of the number of AS-level adversaries that could disconnect them from all relay nodes. While FIBRE is slightly better than Falcon, SABRE outperforms both.

D. SABRE efficiency compared to hosting all clients in /24s

We now compare SABRE to the most effective countermeasure against routing attacks: hosting *all* Bitcoin clients in /24 prefixes [17]. While effective, this countermeasure is also highly impractical as it requires ISP cooperation in addition to an increase in the size of the routing tables Internet-wide. We found that SABRE offers comparable level of protection against network-wide and targeted attacks while being easily deployable.

The comparison between the two approaches is not straightforward as SABRE protects the network even if the attacker has already partitioned the Bitcoin Peer-to-Peer network while the other approach (hosting clients in /24) aims at securing the Peer-to-Peer network itself. We compare them against the same metrics we used previously in Section VI-B, namely offered protection against partition attacks and effectiveness in protecting individual nodes. In the following we describe how we calculated those metrics for the all-/24 approach, before presenting our results.

First, we estimate the size of different partitions and the number of AS-level adversaries that could achieve those assuming all clients are hosted in /24 prefixes. In particular, we find the AS-level adversaries that would be able to isolate a considerable fraction of Bitcoin clients using same-prefix advertisements only. To that end, we run a search on the AS-level topology graph staring from each AS with Bitcoin clients X and following order of descending path preference (as described in Section IV-C), namely more economically preferred paths for X are visited first. All ASes that are traversed by the search before another AS with Bitcoin clients are able to isolate X from the Bitcoin network. Indeed, this calculation gives only a lower bound with respect to the possible partitions, i.e. hosting all clients in /24 prefixes might offer less security than what we computed. Finally we

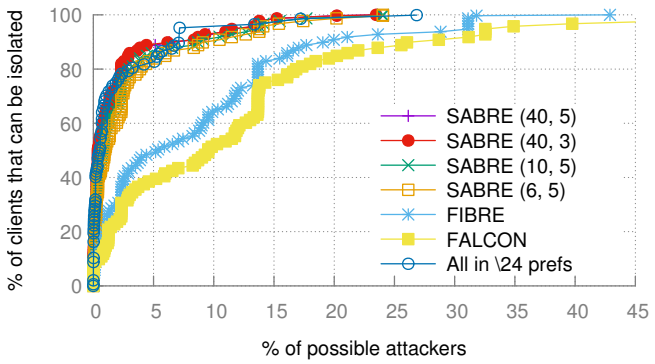


Fig. 11: Falcon does not protect many clients as it is centralized to only two ASes. SABRE performs on-par with hosting all clients in /24 while being deployable (Tie breaks in favor of the attacker)

compare our findings to SABRE's. Our results are summarized in Fig. 10. Indeed, hosting all clients in /24 prefixes would secure the Bitcoin Network better than SABRE, as partitions larger than 20% would be possible for only 0.016% of ASes.

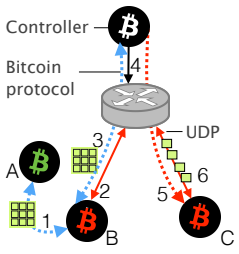
Second, in order to calculate how many attackers can successfully isolate individual Bitcoin clients assuming that all clients are hosted in /24 prefixes, we looked at the ASes that are able to divert traffic from each of those clients to all others in the network. We compare our findings to SABRE's. The results are included in Fig. 11. The two approaches show similar protection levels with SABRE being slightly better at times. This is because SABRE can place relays in any AS in the Internet, while the alternative countermeasure is limited to the actual distribution of Bitcoin clients.

VII. Software/Hardware Co-design Feasibility

We validated the feasibility of our co-design by testing it in practice using regular and modified Bitcoin clients. In this section, we showcase that: *(i)* a programmable switch can seamlessly talk to a Bitcoin client without any software interaction; and that *(ii)* the data-plane memory footprint is low compared to the on-chip memory available in today's programmable switches.

Implementation and testbed Both the controller and the clients are implemented as extensions of the default Bitcoin client version 0.16. The former containing ~650 added or modified lines of C++ code and the latter ~680 lines. The switch is implemented in ~900 lines of P4 code. Our prototype runs on Mininet [40] and uses the publicly available P4 behavioral model (BMV2) [14] to emulate the switch. Our testbed (see Fig. 12) is composed of three clients A, B, C along with a relay node consisting of a switch and a controller. Nodes B, C (shown in red) are modified and are connected to SABRE, while node A (shown in green) is an unmodified Bitcoin client.

Timing We walk through the life of a block that was mined in the Bitcoin network and sent by the unmodified client A to B. The latter will advertise the new block to the switch



	Time	Description
1	0	Node B gets block from node A
2	69ms	Node B sends ADV to switch
3	14ms	Node B connects to Controller
4	7s	Controller updates the switch
5	1.6ms	Controller advertises block to C
6	7.22s	Node C gets block in segments

Fig. 12: A block can be successfully transmitted from node A to node C via the SABRE after it has been validated by the controller.

which will allow node B to connect directly to the controller and transmit it. The controller will then update the memory of the switch and will advertise the block to the connected peers (e.g., C). Next, node C will request and receive the block in segments. The main steps of this procedure are listed in Fig. 12 which describes each step and the time it required in our prototype implementation. The most time-consuming operations are updating the switch and transmitting the block, taking ~7s each. These high delays are due to the fact that we relied on a software-based P4 switch. In practice, the only actual bottleneck in a hardware implementation would be the uplink of the relay nodes.

Memory requirements We analytically calculated the memory required for each of the components in the switch taking into consideration the expected load. Table I summarizes our results. It contains the name of the component and its capacity, i.e., the number of elements that can be added such that the false positive rate listed in the third column is not exceeded. We found that the cumulative memory needed does not exceed 5MB which is well within the limitations of today's programmable switches. The most memory-demanding component is the *Blacklist* for which we budget 1 million entries. This is necessary to allow for mitigating DDoS attacks. In contrast, the components devoted to benign operations are less memory-demanding since the number of legitimate clients is significantly less. For instance, we only reserve space for $100k$ clients in the *PeerList* and 100 for the *Whitelist*; both require less than 1MB. Observe that Bloom filters for regular clients have a lower false positive rate than the *Blacklist*. This enables to serve already connected clients even if the switch is under such an extreme DDoS attack that the *Blacklist* is flooded. We do not list the requirement of the *SentLimit* component as they are negligible [49]. Finally, the memory needed for storing the latest block as well as for keeping all known hashes takes about 1MB each.

VIII. Deployability & Incentives

In this section, we show that SABRE is both practical and partially deployable. While a full deployment of SABRE can protect Bitcoin as a whole, partially deploying SABRE is less expensive and offers gains to early adopters (even individuals).

A. Full deployment

A complete deployment of SABRE requires: *(i)* hosting relays in particular ASes; *(ii)* equipping relay locations with

Component	*Items*	*False Positive*	*Memory*
BlackList:	1000000	0.001	1.80MB
WhiteList:	100	0.0001	239.75B
HashMem:	518823	0.0001	1.24MB
PeerList:	100000	0.0001	479.25K
blockMem:	1	-	1.0MB

TABLE I: The memory used in the P4 switch is always <5MB

specialized hardware; and *(iii)* incentivizing a third party to build and maintain the infrastructure. In the following, we explain why each of those requirements is practical.

First, we observe that a large number of ASes that can host SABRE relays are cloud providers, CDNs, IXPs, large ISPs, or Software-as-a-Service (SaaS) providers. This should come as no surprise as such ASes are actively trying to establish as many peering connections as possible to improve their services. Deploying SABRE nodes in such ASes is practical as they already sell online services or are research-friendly (IXPs [33], [32]). Moreover, even if some eligible ASes do not consent to host SABRE nodes, the effectiveness of SABRE will not be significantly affected as: *(i)* SABRE only requires few nodes to be useful (as little as 6 ASes, see Section VI); and *(ii)* there are more than 2000 possible locations for hosting ASes. In short, no candidate AS is irreplaceable.

Second, we argue that cloud providers could start renting out hardware-accelerated computing instances with programmable network data planes. Actually, some cloud providers already allow clients to rent advanced hardware resources. For instance, Amazon EC2 offers the possibility to connect computing instances with field-programmable gate arrays [6]. That said, a pure software-based implementation of SABRE would still protect the Bitcoin network from routing attacks, leaving DDoS protection to the default mechanism operated by each AS. As described above, such a software-based implementation could be readily deployed as it only requires the possibility to host virtual machines in candidate ASes.

Third, we argue that the possible monetary losses induced by routing attacks [17] create business incentives for one or more entities to deploy and maintain relay nodes. We observe that such incentives are similar to the ones behind existing relay networks such as FIBRE [2] and Falcon [3].

Observe also that deploying SABRE does not need to be approved by the community as a whole. Indeed, a regular client can connect to a SABRE node via a single lightweight UDP connections by independently upgrading its code. Thus, the notoriously slow-moving Bitcoin community cannot be an obstacle to SABRE's deployment.

B. Partial deployment

While feasible, fully deploying SABRE is time-consuming and requires multiple parties to collaborate and possibly share costs. Luckily, SABRE can also be partially deployed, e.g. by a mining pool who wishes to protect itself from routing attacks or by existing relay networks that wish to improve their poor VI-C protection against routing attacks.

By deploying SABRE at a low-budget, a single mining pool can secure the propagation of its own blocks and the reception of new mined ones, even while the Bitcoin peer-peer-network is under a severe routing attack. As an illustration, such a deployment of 6 SABRE nodes would only cost 500$/month (considering the current AWS pricing policy [11]). This cost is: (i) well within the financial capabilities of a single mining pool; (ii) entirely justified given the possible losses that a pool could incur upon a successful routing attack (e.g. an orphan block is a 150K loss [13]). Note that the above deployment does not require special hardware to scale as the pool is only interested in serving its own Bitcoin clients (i.e. its gateways) rather than any possible Bitcoin client that might wish to benefit from the system.

Of course, as multiple pools start to build SABRE networks, they can collaborate and share costs, henceforth incentivizing the deployment of larger SABRE networks (possibly using hardware-accelerated instances which are also already sold by cloud providers [13]) that could protect more clients.

At the same time, existing relay networks such as FIBRE and Falcon can also have significant gains by partially deploying SABRE as its relay location algorithm is orthogonal to their approaches. To do so, existing relays need to independently relocate their servers accordingly to SABRE's network design.

IX. Discussions

In this section we discuss high-level concerns often raised against SABRE, focusing on its potential impact on the Bitcoin ecosystem and its applicability to other Blockchain systems.

Isn't SABRE violating Bitcoin decentralization premises? *No*, for three main reasons. First, it acts alongside Bitcoin peer-to-peer network and does not intend to replace it. Instead, SABRE enhances the connectivity of the Bitcoin network, henceforth reducing its attack surface. Second, SABRE does not need to be centralized: multiple SABRE-like systems can easily co-exist, each belonging to a different entity. Third, SABRE has the potential to allow less well-connected miners to get their fair-share out of the block-rewards making it less likely for others to engineer block races.

We observe that existing relays such as FIBRE [2] and Falcon [3] are small and controlled by a single entity. Yet, these characteristics did not prevent them from having positive impact on Bitcoin by decreasing the orphan rate.

Why focusing on Bitcoin? We focus on Bitcoin as opposed to other cryptocurrencies (e.g., Ripple [16], Ethereum [12]) for three main reasons. First, the Bitcoin network is extensively studied [43], [23], [45] and the effectiveness of routing attacks against it is well-understood [17]. In contrast, more sophisticated Blockchain systems (e.g., Bitcoin-NG [25], Ouroboros [37], OmniLedger [39], Algorand [28]) are not yet deployed at large scale. Thus, their exact routing characteristics are unknown. Second, Bitcoin remains the most widely used cryptocurrency making its security vital for more users.

Can SABRE protect other blockchains? *Yes*. Although SABRE focuses on Bitcoin, its network and node design principles are general and apply to other blockchain systems.

SABRE network design can help blockchain systems mitigating partition attacks [17]. Partition attacks are a threat to any blockchain systems depending on Internet connectivity, including permissioned and/or encrypted ones. SABRE allows nodes to exchange information even if a malicious AS-level adversary hijacks and drops traffic among them. In fact, the properties upon which the SABRE network is built can also be used by miners to interconnect and/or host their mining power, or by new blockchains to place their nodes. Note that SABRE network design would also be useful to advanced blockchain systems such as ByzCoin [38] and OmniLedger [39] which currently mitigate the effects of partition attacks by freezing commits. SABRE would allow them to retain liveness instead of waiting for the attack to be resolved.

In contrast to its network design, SABRE node's design is more Bitcoin-specific. For example, blockchains whose traffic is encrypted cannot be served from a programmable network device. Even so, SABRE's node design exhibits two key properties that most blockchain systems can leverage. First, blockchain systems tend to be communication-heavy (due to the need to reach consensus) meaning that the use of programmable switches can increase the throughput by offloading communication burden to the hardware. Second, most popular items are predictable, as most requests target the latest mined content, making SABRE-like caching strategies very effective.

X. Related work

Using P4 switches as cache Previous works have used programmable network devices to cache values including Netcache [35] and NetChain [34]. Netcache uses Tofino switches [7] as a cache for key-value stores, to deal with skewed requests in memcached applications. Similarly, NetChain [34] caches key-values stores in switches to boost Paxos consensus protocols used in data centers to coordinate servers. In SABRE, we also rely on switches to cache information (here, blocks) but also to distinguish maliscious clients and to filter incomming information.

BGP security Many proposals have been proposed over the years to reduce or prevent routing attacks. We distinguish two approaches: origin validation and path validation. Origin validation [41] relies on RPKI [20], a X.509-based hierarchy mapping IP prefixes to ASes, to enable the routers to filter BGP advertisements originated from unauthorized ASes. Path validation [42] secures BGP by adding cryptographic signatures to the BGP messages. It allows the recipient of an announcement to cryptographically validate that: (i) the origin AS was authorized to announce the IP prefix; *and* (ii) that the list of ASes through which the announcement passed were indeed those which each of the intermediate AS intended. Unfortunately, none of these proposals have been widely deployed, leaving the Internet still vulnerable to routing attacks [29]. In contrast, SABRE enables to secure Bitcoin against routing attacks today, without requiring all ASes to agree or change their practices .

Routing attacks on ToR Extensive work has been done on routing attacks on ToR [52] and how these can be circumvented [50], [46] [51]. There are three key differences between the ToR relay network and the Bitcoin network that change the spectrum of possible attacks and countermeasures. First, in order to protect the Bitcoin system we need to keep the network connected as opposed to preserving the privacy of every single connection for ToR. As such, we can use redundancy to protect Bitcoin clients, by connecting them to multiple SABRE relays such that there is no AS that can effectively hijack all connections. Second, counter-measures against routing attacks on ToR are limited to avoiding routes that might be affected by BGP hijacks, while SABRE is built to avoid the chance of an attacker to be able to divert it in the first place. This is possible because Bitcoin clients have no preference with respect to who to connect to as they can get the same information from almost any peer. Third, countermeasures against routing attacks on ToR do not deal with the case that the client itself is hijacked.

Multicast protocols Mbone [24], was designed to multicast live videos and music streams in the Internet, where many routers do not support IP multicast. Using tunnelling, Mbone traffic can stay under the radar of those routers. Despite its novelty and usefulness, this network does not take into consideration whether the used paths can be hijacked and does not deal with maliciously increased load. Finally, systems such Splitstream [21] that aim to reduce the load per node, require a fixed set of participants and a certain structure among them which would limit the openness of our network (regular clients cannot easily come and go).

XI. Conclusion

We presented SABRE, a relay network aimed at securing Bitcoin against routing attacks. The key insight behind SABRE is to position the relay nodes in secured locations, preventing AS-level attackers from diverting intra-relay communications and reducing their ability to divert traffic destined to the relay clients. To protect the nodes themselves, SABRE leverages a hardware/software co-design (leveraging programmable data planes) to perform most of the relay operations in hardware. We fully implemented SABRE and demonstrated its effectiveness in protecting Bitcoin, with as little as 6 relay nodes.

Acknowledgments

We thank the members of the Networked Systems Group at ETH Zurich for their valuable feedback. We also thanks the anonymous reviewers for their comments and guidance. This work was supported by a Swiss National Science Foundation Grant ("Data-Driven Internet Routing", #200021-175525).

References

[1] "CAIDA Macroscopic Internet Topology Data Kit." https://www.caida.org/data/internet-topology-data-kit/.

[2] "Fast Internet Bitcoin Relay Engine," http://bitcoinfibre.org/.

[3] "A fast new bitcoin backbone relay network," https://www.falcon-net.org.

[4] "RIPE RIS Raw Data," https://www.ripe.net/data-tools/stats/ris/ris-raw-data.

[5] "The Bitcoin Relay Network," http://bitcoinrelaynetwork.org/.

[6] "Amazon EC2 F1 Instances are now available in AWS GovCloud," 2017, https://aws.amazon.com/about-aws/whats-new/2017/11/amazon-ec2-f1-instances-are-now-available-in-aws-govcloud--us/.

[7] "Barefoot Tofino Switches: The Technology," 2018, https://www.barefootnetworks.com/technology/.

[8] "Bitnodes api," 2018, https://bitnodes.earn.com/api/.

[9] "Bitnodes Statistics," 2018, https://bitnodes.earn.com/.

[10] "CIDR report," 2018, http://www.cidr-report.org/as2.0/.

[11] "EC2Instances.info Easy Amazon EC2 Instance Comparison." 2018, https://www.ec2instances.info/?selected=f1.2xlarge,f1.4xlarge.

[12] "Etherium," 2018, https://www.ethereum.org/.

[13] "How Bitcoin Mining/Block rewards work." 2018, https://www.anythingcrypto.com/guides/bitcoin-mining-block-rewards-2018.

[14] "P4 Behavioral Model," 2018, .https://github.com/p4lang.

[15] "P416 Language Specification," 2018, https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html.

[16] "Ripple," 2018, https://ripple.com/.

[17] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017.

[18] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in Bitcoin P2P network," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, 2014, pp. 15–29. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660379

[19] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[20] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol," RFC 6810, Jan. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6810.txt

[21] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 298–313.

[22] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 205–216. [Online]. Available: http://doi.acm.org/10.1145/2463676.2465323

[23] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[24] H. Eriksson, "Mbone: The multicast backbone," *Communications of the ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[25] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol." in *NSDI*, 2016, pp. 45–59.

[26] U. Feige, "A threshold of ln n for approximating set cover," *J. ACM*, vol. 45, no. 4, pp. 634–652, Jul. 1998. [Online]. Available: http://doi.acm.org/10.1145/285055.285059

[27] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 6, pp. 681–692, 2001.

[28] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[29] S. Goldberg, "Why is it taking so long to secure internet routing?" *Commun. ACM*, vol. 57, pp. 56–63, 2014.

[30] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford, "How secure are secure interdomain routing protocols," in *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4. ACM, 2010, pp. 87–98.

[31] ——, "How Secure Are Secure Interdomain Routing Protocols," in *SIGCOMM*, 2010.

[32] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "An Industrial-Scale Software Defined Internet Exchange Point," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX, 2016.

[33] A. Gupta, L. Vanbever, M. Shahbaz, S. Donovan, R. Clark, N. Feamster, J. Rexford, and S. Shenker, "SDX: A Software Defined Internet Exchange," in *Proceedings of the 2014 ACM SIGCOMM Conference (SIGCOMM'14)*. ACM, 2014.

[34] X. Jin, X. Li, H. Zhang, N. Foster, J. Lee, R. Soulé, C. Kim, and I. Stoica, "Netchain: Scale-free sub-rtt coordination," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, 2018, pp. 35–49. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/jin

[35] X. Jin, X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, and I. Stoica, "Netcache: Balancing key-value stores with fast in-network caching," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 121–136.

[36] J. Karlin, S. Forrest, and J. Rexford, "Pretty good bgp: Improving bgp by cautiously adopting routes," in *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, ser. ICNP '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 290–299. [Online]. Available: http://dx.doi.org/10.1109/ICNP.2006.320179

[37] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[38] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 279–296.

[39] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger." *IACR Cryptology ePrint Archive*, vol. 2017, p. 406, 2017.

[40] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[41] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480, Feb. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6480.txt

[42] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8205.txt

[43] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," *et al.*, 2015.

[44] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[45] T. Neudecker, P. Andelfinger, and H. Hartenstein, "Timing analysis for inferring the topology of the bitcoin peer-to-peer network," in *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress, 2016 Intl IEEE Conferences*. IEEE, 2016, pp. 358–367.

[46] R. Nithyanand, R. Singh, S. Cho, and P. Gill, "Holding all the ases: Identifying and circumventing the pitfalls of as-aware tor client design," *arXiv preprint arXiv:1605.03596*, 2016.

[47] A. Pilosov and T. Kapela, "Stealing The Internet. An Internet-Scale Man In The Middle Attack." DEFCON 16.

[48] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

[49] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*. ACM, 2017, pp. 164–176.

[50] O. Starov, R. Nithyanand, A. Zair, P. Gill, and M. Schapira, "Measuring and mitigating as-level adversaries against tor," *arXiv preprint arXiv:1505.05173*, 2016.

[51] Y. Sun, A. Edmundson, N. Feamster, M. Chiang, and P. Mittal, "Counter-raptor: Safeguarding tor against active routing attacks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 977–992.

[52] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "Raptor: Routing attacks on privacy in tor." in *USENIX Security Symposium*, 2015, pp. 271–286.

APPENDIX

A. Algorithms

Below we include the pseudocode of the two main algorithms described in the paper in Section IV-B and Section IV-C respectively.

Algorithm 1 Find a set of ASes to locate relays.

function LOCATERELAYS(R, R_scens, N, k) ▷ finds N k-connected relay ASes
▷ R_scens : scenarios that each candidate relay AS in R protects against
  $R' \leftarrow \{\}$ ▷ ASes to host relay nodes
  $R'\_scens \leftarrow \{\}$ ▷ scenarios relays in R' protect against
  while R'.length < N **do**
    $Rs \leftarrow \{r : r \in R \setminus R' \ s.t. G[R' \cup r]$ is k-connected$\}$ ▷ candidate relays
    $R'.add($ FINDNEXT (Rs, R_scens, R'_scens) $)$
  end while
  return R'
end function
function FINDNEXT(Rs, R_scens, R'_scens) ▷ finds best relay to add to R'
  $best\_r \leftarrow None$; $best\_scens \leftarrow \{\}$;
  for r in Rs **do**
    $tmp\_scens \leftarrow R'\_scens \cup R\_scens[r]$
    if $best\_scens.effect < tmp\_scens.effect$ **then**
      $best\_scens \leftarrow tmp\_scens$
      $best\_r \leftarrow r$
    end if
  end for
  $R'\_scens.add(R\_scens$[best_r])
  return $best\_r$
end function

Algorithm 2 Compare two paths based on preference.

function MOREPREFERRED($pathA$, $pathB$) ▷ returns 0 if $pathA$ is more preferred and 1 otherwise
  while pathA & pathB & hopA.pick==hopB.pick **do**
    hopA ← pathA.pop()
    hopB ← pathB.pop()
  end while ▷ Traverse until the two paths
  if $hopA.type \neq hopB.type$ **then**
    switch (hopA.type , hopB.type) **do**
      case (customer, peer)
        return 0
      case (customer, provider)
        return 0
      case (peer, provider)
        return 0
      case (peer, customer)
        return 1
      case (provider, customer)
        return 0
      case (provider, peer)
        return 1
  else
    if $len(pathA) = len(pathB)$ **then**
      return 1 ▷ In case of a tie, we prefer path B.
    else
      return argmin(len(pathA), len(pathB))
    end if
  end if
end function

B. Results with ties against the attacker

Although SABRE significantly improves the security of Bitcoin against routing attacks the exact partition sizes and number of vulnerable clients depend on the tie-breaking decisions, namely which path is chosen in cases that the competing routes are equivalent economically and length-wise. In Section VII, we assumed that the tie always breaks for the attacker. In the following, we include the same analysis only now assuming that the tie-breaking favors the legitimate destination.

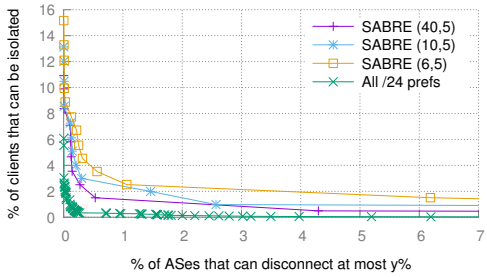


Fig. 13: When tie breaks in favor of the legitimate destinations: a SABRE of only 6 relays that are fully connected prevents all attackers from isolating more than 16%.

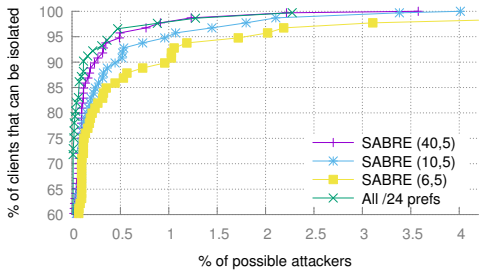


Fig. 14: When tie breaks in favor of the legitimate destinations: a SABRE of 10 5-connected relays protects 95% of the clients from 99.5% of the AS level adversaries.

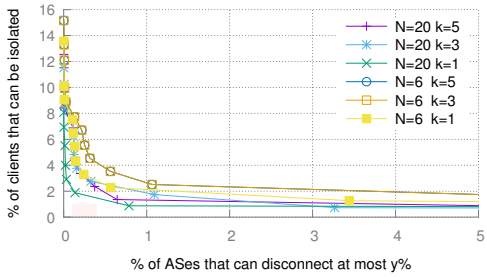


Fig. 15: When tie breaks in favor of the legitimate destinations: the largest possible partition by any attacker is 14% for a SABRE of 6 relays that is 5-connected.

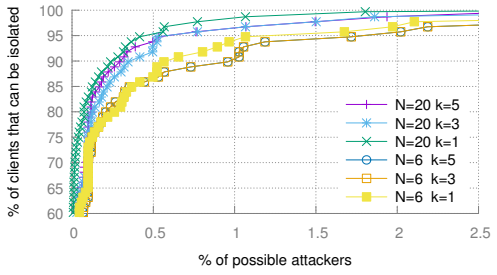


Fig. 16: When tie breaks in favor of the legitimate destinations: a SABRE of 20 relays that are 1-connected can secure 100% of the clients against more than 98% attackers.