Anonymous Addresses for Efficient and Resilient Routing in F2F Overlays

Stefanie Roos, Martin Beck, Thorsten Strufe
TU Dresden, Privacy and IT Security
{stefanie.roos,martin.beck1,thorsten.strufe}@tu-dresden.de

***Abstract*—Friend-to-friend (F2F) overlays, which restrict direct communication to mutually trusted parties, are a promising substrate for privacy-preserving communication due to their inherent membership-concealment and Sybil-resistance. Yet, existing F2F overlays suffer from a low performance, are vulnerable to denial-of-service attacks, or fail to provide anonymity. In particular, greedy embeddings allow highly efficient communication in arbitrary connectivity-restricted overlays but require communicating parties to reveal their identity. In this paper, we present a privacy-preserving routing scheme for greedy embeddings based on anonymous return addresses rather than identifying node coordinates. We show that the return addresses allow plausible deniability. Furthermore, we enhance the routing's resilience by using multiple embeddings and propose a method for efficient content addressing. Our extensive simulation study on real-world data indicates that our approach is highly efficient and effectively mitigates failures as well as powerful denial-of-service attacks.**

I. Introduction

Anonymous and censorship-resistant communication is a prerequisite for freedom of speech. Threats to this essential human right have recently emerged even in western countries, the fear of surveillance of personal communication has lead to observable self-censorship[1]. The inherent vulnerability of publicly known servers to sabotage necessitates decentralized solutions for anonymous communication and content distribution. Yet, the openness of such systems presents a vulnerability in itself, enabling attackers to infiltrate the system with a large number of forged participants, as observed in attacks on the Tor [1] network in 2014[2].

F2F overlays reduce disclosure to strangers by restricting connectivity to participants who share a mutual trust relationship in the real world. Hence, adversaries need to resort to social engineering attacks to infiltrate the network. Large-scale privacy-preserving communication in F2F overlays still requires additional measures to achieve anonymity, resilience to failures and attacks, and efficiency. Multiple studies indeed have shown that deployed F2F overlays as well as proposed alternatives are highly inefficient and vulnerable to attacks [2]–[4].

Greedy network embeddings, as proposed in e.g. [5], [6], are a potential solution. They allow for efficient greedy routing with local knowledge on arbitrary graphs. For this purpose, they first construct a spanning tree of the network and then assign coordinates based on a node's position in the tree. However, contacting an untrusted user requires knowledge of its coordinate in the network. Though only direct neighbors can map the embedding coordinate to a real-world identity, arbitrary participants can reconstruct the social graph based on the coordinates that are revealed through the protocols. Participants then can then be identified from the social graph structure [7] and potentially tracked.

Our first requirement for F2F overlays thus is that addressing and routing protocols are modified to achieve efficiency and anonymity. Second, in the face of well-known attacks, the resilience of such embeddings to both failures and attacks have to be increased dramatically. Third, efficient content storage and retrieval requires a suitable content addressing scheme that exploits the network embeddings.

Our solution meets the above requirements by i) introducing anonymous return addresses to provide receiver anonymity, ii) leveraging multiple embeddings and backtracking during routing to increase resilience, and iii) constructing a virtual overlay that exploits the network embeddings and thus avoids the enormous stabilization costs of previous approaches.

We evaluate our solution both by a formal security and performance analysis and an extensive simulation study. The performance analysis shows that our scheme achieves scalable and efficient routing as well as efficient self-stabilization in the presence of dynamic arrival and departure of participants. In the security analysis, we prove that our return addresses indeed provide receiver anonymity in the form of plausible deniability. Furthermore, our simulation study indicates a high robustness and attack-resistance in comparison to state-of-the-approaches. We therefore believe that our F2F overlay is the first to offer anonymity, resilience, and efficiency.

II. Related Work

Here, we describe the state-of-the-art with regard to routing and content discovery in F2F overlays.

Unstructured approaches utilize *flooding*, e.g. in Turtle [8], or *probabilistic forwarding* e.g. in OneSwarm [9]. GnuNet attempts to combine random walks with deterministic routing [10]. These overlays focus locating content rather than individual nodes. Due to the replication of content, the content can indeed be located, but efficient communication between two uniquely defined entities is not possible.

Virtual overlays address the problem of establishing an overlay despite the restricted connectivity by replacing overlay

[1]http://www.theguardian.com/commentisfree/2013/jun/17/chilling-effect-nsa-surveillance-internet

[2]https://blog.torproject.org/category/tags/attacks

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

links with tunnels of trusted nodes. So, efficient tunnel discovery and maintenance is a main concern given the inherent network dynamics: Vasserman et al. [3] suggest flooding the network for discovering adequate overlay neighbors, thus creating a large overhead. In contrast, X-Vine [11] leverages the overlay routing by concatenating previously existing tunnels to a new one, thus entailing a increase of the average tunnel length and hence routing costs over time [4].

In contrast, *network embeddings* assign coordinates that allow efficient routing to nodes. For example, the F2F mode of Freenet relies on a network embedding. However, results indicate that the embedding is lacking both with regard to routing efficiency [3] and attack resilience [2]. Hoefer et al. [12] propose highly efficient *greedy embeddings*. However, their approach reveals the identity of the communicating parties and fails to consider resilience. Furthermore, their proposed scheme for content addressing maps the majority of content keys to the same central node.

In summary, network embeddings are the only existing approach providing a high efficiency at acceptable maintenance costs. However, achieving receiver anonymity, resilience, and suitably content addressing is an unsolved highly challenging problem.

III. Network Embeddings

Our solution builds upon previous work in the area of network embeddings, which assign coordinates to nodes with the goal of structuring networks. In the following, let $G = (V, E)$ be a network and $(\mathbf{X}, \delta_X)$ be a metric space with a distance δ_X. A network embedding is defined as a function $id : V \to \mathbf{X}$ assigning each node a coordinate. The problem of enabling routing in a connectivity-restricted network has been addressed by the design of *greedy embeddings*. Greedy embeddings [13] are coordinate assignments, such that for any source-destination pair $(s, t) \in V \times V$ with $s \neq t$, a neighbor u of s exists such that $\delta_X(u, t) < \delta_X(s, t)$. We say that u is closer to t than s with regard to δ_X. As a consequence, straight-forward greedy routing is guaranteed to find a route from s to t.

Though there exists a multitude of greedy embedding algorithms, they all follow the same four abstract steps: i) Construct a spanning tree T, ii) Each internal node in T enumerates its children, iii) The root receives a predefined coordinate, iv) Children derive their coordinate from the parent's coordinate and the enumeration index assigned by the parent (e.g. [5], [6], [14], [15]). The coordinates are then distributed such that the embedding of the spanning tree is greedy, as specified for the PIE embedding below. Subsequent to the coordinate assignment, nodes consider all neighbors, including those that are neither parent nor child, for the routing. So, routing is not restricted to tree edges. We call non-tree edges *shortcuts* because they allow for a faster reduction of the distance and shorter routes than predicted by the distance in the tree.

Though embedding algorithms generally rely on a spanning tree and assign coordinates according to the tree structure, the nature of the assigned coordinates is highly diverse: Embeddings into hyperbolic space such as [5], [14], [15] allow embedding in low-dimensional spaces. However, proposed hyperbolic embeddings are extremely complex and do not scale with regard to the number of bits required for coordinate representation [15]. Custom-metric approaches have been designed to overcome these shortcomings. The custom-metric embedding PIE [6] assigns an empty vector as the root coordinate. Child coordinates are then derived from the parent coordinate by concatenating the parent coordinate with the index assigned to the child by the parent, potentially weighted with the cost of the parent-child edge if such weights are given. In this manner, a node s's coordinate represents the route from the root to u. Consequently, the distance δ_X is given by the hop distance of two nodes in the tree. Whereas routing in greedy embeddings is highly efficient in comparison to non-greedy embeddings [12], neither anonymity nor resilience has been considered in a suitably manner.

IV. Adversary Model

We aim to realize efficient routing and resource discovery in F2F overlays making use of network embeddings. At the same time, we need to provide receiver anonymity, resilience, and content addressing. Thus, two attack goals are considered in our adversary model, namely i) identifying communicating parties, i.e., the sender and receiver of a message and ii) blocking undesired communication, for instance by a denial-of-service (DoS).

As for the attacker's capabilities, we assume a local, active, internal, and possibly colluding attacker, who is able to drop and manipulate the messages she receives. A global passive attacker is disregarded as steganographic techniques can be applied to hide the F2F traffic (cf. [16]). Furthermore, the adversary is computationally bounded to polynomial time, which in particular prevents her from breaking computationally-secure cryptographic primitives. However, in order to protect against powerful adversaries, such as government bodies, we assume that the adversary is computationally strong, possibly stronger than all honest users combined. Regarding the number of malicious nodes and their impact, the adversary can control an arbitrary number of colluding *Sybil* nodes in the network. She is unable to observe the complete topology and can only establish a limited number of connections to honest nodes, as the seeming real-world trust relationships have to be social-engineered. We particularly assume that i) an adversary cannot be certain that she knows all neighbors of a node, and ii) the total number of connections to honest nodes is small, meaning bound by $\mathcal{O}(\sqrt{n})$ in agreement with related studies [17].

Based on the above capabilities and goals, the adversary may apply the following strategies. To break the *anonymity* of communicating parties, the adversary learns from messages that are passed through her, either directly from the content of the messages or indirectly from more sophisticated attacks, such as timing analyses.

As for the routing *resilience*, we focus on adversaries dropping messages. In order to maximize its impact, we assume the

adversary to precede her denial-of-service attack by gaining an influential position in the network, employing, e.g., a black hole attack [18]. To attack the network embeddings, the adversary A is assumed to interrupt the communication from her descendants in the spanning tree to other parts of the tree by dropping all messages forwarded to A. Though the use of shortcuts to different parts of the tree mitigates the impact of the above attack, the resilience of a single network embedding to such attacks may be insufficient, especially if the adversary manages to sabotage the spanning tree construction. We do not specifically consider pollution and local eclipse attacks in this paper. Although they present a threat for F2F networks, they have been addressed in various publications (cf. [19]), which can be applied to our contribution with few modifications.

V. Design

Our main contribution lies in proposing multiple greedy embeddings with anonymous return addresses and a virtual overlay on top of the embeddings. Throughout this section, let b be a sufficiently large integer, $PRNG$ a pseudo-random number generator with values in $\mathbb{Z}_2^b$, and $h : \{0,1\}^* \to H$ a cryptographically secure hash function.

A. Tree Construction and Stabilization

In this section, we show how we construct and stabilize γ parallel spanning trees. In the next section, we then describe how to assign coordinates on the basis of these trees. We want to increase the robustness and censorship-resistance by using multiple trees. In order to ensure that the trees indeed offer different routes, our algorithm encourages nodes to select different parents in each tree if possible.

Tree Construction : We divide the construction of a tree into two phases i) selecting the root, and ii) building the tree starting from the root. We can apply [20] for the root election, which achieves a communication complexity of $\mathcal{O}(n \log n)$. Our own contribution lies in ii) the tree construction after the root node has been chosen.

We now shortly describe the idea of our algorithm. Due to space constraints, we limit our description of the algorithm to an informal characterization. Our technical report includes the pseudocode and expands on the role of the essential parameter q [21]. A node u that is not the root receives messages, called invitations in the following, from its neighbors when they join a tree and hence become potential parent nodes. There are two questions to consider when designing an algorithm governing u's reaction to such messages. First, u has to decide if and when it accepts an invitation. Second, u has to select one invitation in the presence of multiple invitations.

For the second question, u always prefers invitation from nodes that have been their parent in less trees with the goal of constructing different trees and increasing the overall number of possible routes. Increasing the number of routes allows the use of alternative routes if the request cannot be routed along the preferred route due to a failed or malicious node. If two neighbors are parents in the same number of trees, u can either select one randomly or prefer the parent closer to the root. Choosing a random parent reduces the impact of nodes close to the root but is likely to lead to longer routes and thus a lower efficiency.

Coming back to the first question of if and when u accepts invitations, u always accepts an invitation of a neighbor v that is not yet a parent of u in any tree. In contrast, if v is already a parent, u might wait for the invitation of a different neighbor. However, it is unclear if it is possible for all neighbors of u to ever become a parent. For example, a neighbor of degree 1 is only a parent if it is the root. In order to overcome this dilemma, u periodically probabilistically decides if it should accept v's invitation or wait for another invitation. The probability of accepting one invitation in the absence of any invitations from preferred neighbors, i.e., neighbors that are not yet parents, is q. Note that u accepts at most one invitation per round, i.e., per periodic consideration of parent nodes. So, u eventually accepts an invitation but does provide alternative parents the chance to send an invitation.

Stabilization : Now, we consider the stabilization of the trees when nodes join and leave. Stabilizing the trees efficiently, i.e., repairing them locally rather than reconstructing the complete tree whenever the topology changes, is essential for efficiency. Joining nodes can be integrated in a straight-forward manner by connecting to their neighbors as children, again trying to maximize the diversity of the parents. For this purpose, nodes record the time, i.e., the round in our abstract time model, they joined the tree. Now, when a new node u joins, it requests its neighbors' coordinates and these timestamps for all trees. Based on this information, u can simulate the tree construction locally, ensuring that its expected depth in the tree is unaffected by its delayed join. When a node departs, all its children have to choose a different parent and inform their descendants of the change. In order to prevent a complete subtree from being relocated at an increased depth, the descendants may also select a different parent. The selection of the new parent again the above algorithm but only locally re-establishes the trees affected by the node departure. We formally prove that the above stabilization algorithm indeed only introduces only logarithmic complexity in Theorem VI.3.

B. Embedding and Routing

In this section, we show how we assign coordinates in a spanning tree and how to route based on these coordinates. As we want to prevent an attacker from guessing the coordinate of a receiver, we require a certain degree of in-determinism in the coordinate assignment. We thus choose a slightly modified version of the unweighted PIE embedding [6], which we have introduced in Section III. Our proposed routing algorithm corresponds to the greedy routing with backtracking. In addition to the tree distance in [6], we also present a second distance preferring nodes with a long common prefix and thus avoiding routes via nodes close to the root whenever possible. In this manner, we increase robustness and censorship-resistance because the routing algorithm considers alternative routes. In the

following, we subsequently present the embedding algorithm, the distance functions, and the routing with backtracking.

Embedding Algorithm : Embeddings are computed on each of the γ trees independently, so that we only consider one embedding id. The coordinate assignment starts at the root and then spreads successively throughout the tree. After a spanning tree has been established, the root r is assigned an empty vector as a coordinate $id(r) = ()$. In the next step, each child v of the root generates a random b-bit number $a \in \mathbb{Z}_2^b$ such that its coordinate is $id(v) = (a)$. Here, our algorithm differs from the PIE embedding because it uses random rather than consecutive numbers, thus preventing an adversary from guessing the coordinate in an efficient manner. Subsequently, nodes in the tree are assigned coordinates by concatenating their parent's coordinate with a random number. So, upon receiving its parent coordinate $id(p(v)) = (a_1, \ldots, a_{l-1})$, a node v on level l of the tree obtains its coordinate $id(v) = (a_1, \ldots, a_{l-1}, a_l)$ by adding a random b-bit number a_l. The coordinate space is hence given by all vectors consisting of b-bit numbers, i.e., $\mathbf{X} = \{(a_1, \ldots, a_{l-1}, a_l) : l \in \mathbb{N}_0, a_i \in \{0,1\}^b\}$.

Note that the independent random choice of the b-bit number $a \in \mathbb{Z}_2^b$ might lead to two nodes having the same coordinate. Thus, b should be chosen such that the chance of equal coordinates should be negligible. If two children nevertheless select the same coordinate, the parent node can inform one of them to adapt its choice. By moving the choice of the last coordinate element from the parent to the child, we automatically reduce the impact of a malicious parent as it can not determine the complete coordinate of the child.

Distances : We still need to define distances between coordinates in order to apply greedy routing. For this purpose, we consider two distances on $\mathbf{X}$. Both rely on the common prefix length $cpl(x_1, x_2)$ of two vectors x_1 and x_2 and the coordinate length $|x_1|$.

First, we consider the tree distance δ_{TD} from [6], which gives the length of path between the two nodes in the tree, i.e.,

$$\delta_{TD}(x_1, x_2) = |x_1| + |x_2| - 2cpl(x_1, x_2). \tag{1}$$

Secondly, the common prefix length can be used as the determining factor in the distance function, i.e., for a constant L exceeding the length of all node coordinates in the overlay, we define

$$\delta_{CPL}(x_1, x_2) = \begin{cases} L - cpl(x_1, x_2) - \frac{1}{|x_1|+|x_2|+1}, & x_1 \neq x_2 \\ 0, & x_1 = x_2 \end{cases}. \tag{2}$$

The reason for using the common prefix length rather than the actual tree distance is the latter's preference of routes passing nodes close to the root in the tree. In this manner, the nodes on these routes are very influential, so that adversaries can gain a large impact from gaining such a position. In contrast, δ_{CPL} prefers possibly longer routes by always forwarding to a node within the same subtree as the destination and avoids central nodes in the tree.

Greedy Routing in Multiple Embeddings : We route in $1 \leq \tau \leq \gamma$ trees in parallel. More precisely, given a vector of coordinates $(id_1(e), \ldots, id_\gamma(e))$, the sender s selects τ coordinates and sends a request for each of them. s can either select τ embeddings uniformly at random or choose the embeddings so that the distance of the neighbor v_i with the closest coordinate to $id_i(e)$ is minimal. The latter choice might result in shorter routes due to the low distance in the embedding.

The routing processes in each embedding independently. Nodes forward the request to the neighbor with the closest coordinate in the respective embedding. Thus, in order for the nodes on the route to forward the request correctly, the request has to contain both the coordinate $id_i(v)$ and the index i of the embedding.

We optionally increase the robustness and censorship-resistance of the routing algorithm by allowing backtracking if the routing gets stuck in a local minimum of the distance function due to failures or intentional refusal to forward a request. For this purpose, all nodes remember their predecessor on the routing path as well as the neighbors they have forwarded the request to. If all neighbors closer to the target have been considered and have been unable to deliver the request, the node reroutes the request to its predecessor for finding an alternative path. The routing is thus only considered to be failed if the request returns to its source s and cannot be forwarded to any other neighbor. In this manner, all *greedy paths*, i.e., all paths with a monotonously decreasing distance to the target, are found.

This completes the description of the routing and stabilization functionalities. However, up to now, we used identifying coordinates rather than anonymous addresses.

C. Anonymous Return Addresses

In this section, we introduce our address generation algorithm for generating anonymous return addresses that do not reveal the receiver of the request while at the same time allowing the routing to traverse along the same path as for receiver coordinates. For this reason, we call the generated addresses *route preserving (RP) return addresses*. Based on these return addresses, we specify two routing algorithms $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ for routing a request containing a return address.

Return Address Generation : The first step of the return address generation prevents an adversary from identifying coordinates based on their length. A node v pads its coordinate $x = (a_1, \ldots, a_l)$ by adding random elements $a'_{l+1}, \ldots, a'_L$. More precisely, v selects a seed s_{pad} for the pseudo-random number generator $PRNG$ and obtains the padded coordinate $x' = (a'_1, \ldots, a'_l, a'_{l+1}, \ldots, a'_L)$ with $a'_j = a_j$ if $j \leq l$ and $a'_j = PRNG(s_{pad} \oplus j)$ otherwise. In order to ensure that the closest node coordinate to x' is indeed x, v recomputes the padding with a different seed if a'_{l+1} is equal to the $l+1$-th element of a child's coordinate. Afterwards, v chooses a different seed s for the construction of the actual return address and generates $\tilde{k} = PRNG(s) \in \tilde{\mathbb{K}} = \mathbb{Z}_2^b$. v then executes the

local function $hc : \mathbf{X} \to \mathbf{Y} = H^L$ in order to obtain a vector y with elements in H. The i-th element of $y = (d_1, \ldots, d_L)$ is given by

$$d_j = \begin{cases} h(\tilde{k} \oplus a'_1), & j = 1 \\ h(d_{j-1} \oplus a'_j), & j = 2 \ldots L \end{cases}. \quad (3)$$

We call the pair $(y, \tilde{k})$ a *return address*, which can be used to find a route to the node with coordinate x. Before publishing the return address, v adds a MAC $mac(y_i, \mathbb{K}_{MAC}(v)) = h(d_1 || \ldots d_L || \mathbb{K}_{MAC}(v))$ for a private key $\mathbb{K}_{MAC}(v)$ to prevent malicious nodes from faking return addresses and gaining information from potential replies. Last, v publishes the return address $(y, \tilde{k})$ and the MAC.

Routing Algorithms : Now, we determine diversity measures $\delta_{RP-TD} : \mathbf{X} \times \mathbf{Y} \to \mathbb{R}_+$ and $\delta_{RP-CPL} : \mathbf{X} \times \mathbf{Y} \to \mathbb{R}_+$ in order to compare coordinates x and y with regard to δ_{TD} and δ_{CPL}. The diversity measure then assumes the role of the distance δ in the routing algorithm.

In order to define a sensible diversity measure, note that for any coordinate c and return address y corresponding to a coordinate x, we have $cpl(x, c) = cpl(y, hc(c, \tilde{k}))$. We thus can define the diversity measure in terms of the common prefix length in the same manner as the distance. More precisely, for $* \in \{TD, CPL\}$, the diversity $\delta_{RP-*}(y, \tilde{k}, c)$ for of a coordinate c to the return address y is

$$\delta_{RP-*}(y, \tilde{k}, c) = \delta_*(y_i, hc(c, \tilde{k})). \quad (4)$$

In practice, u can increase the efficiency of the computation by only determining $hc(c, \tilde{k})$ up to the first element in which it disagrees with y. Thus, we now have two possible realizations of the routing algorithm, namely $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$. Given the RP return address $(y, \tilde{k})$ of the destination e, $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ forward the message to the neighbor v with the lowest diversity measure $\delta_{RP-TD}(y, \tilde{k}, id(v))$ and $\delta_{RP-CPL}(y, \tilde{k}, id(v))$, respectively. We present a proof that the return addresses are indeed route preserving in our technical report [21].

Up to now, we have only considered route preserving return addresses generated by padding coordinates and applying a hash cascade. Optionally, an additionally layer of symmetric encryption can be added, preventing a node v from deriving the actual length of the common prefix. Rather, v can only determine if a neighbor is closer to the destination than v itself. However, we show the same degree of anonymity for for both algorithms, so that the additional layer does not result in a provably higher level of anonymity. Furthermore, the additional layer reduces the efficiency as nodes select one closer neighbor at random rather than the closest neighbor. For this reason, the advantage of the additional layer is limited, so that we focus on RP return addresses here and defer the further obfuscation of coordinates to our technical report [21].

We prove that our return addresses indeed enable receiver anonymity in Section VI-A.

D. Content Storage

In order to store content, we use a distributed hash table (DHT). As nodes can not communicate directly, they store tree addresses in their routing tables and leverage the tree routing. In this manner, we do not require maintenance-intensive tunnels like [11] and [3]. Note that we only sketch the solution for content storage and retrieval because our focus lies on improving the quality of the greedy embeddings for messaging between nodes.

Nodes establish a DHT by maintaining a routing table of (virtual) overlay connections. The routing table contains entries correspond to a DHT coordinate and corresponding return addresses. Nodes communicate with their virtual neighbors by sending requests in any of the γ embedding.

New routing table entries are added by routing for a suitable virtual overlay key, as done in [11] for the tunnel discovery. However, after the routing terminates, the discovered nodes send back their return addresses rather than taking the routing path as a new tunnel. In this manner, the length of routes between virtual overlay neighbors only depends on the embeddings and does not increase over time. The exact nature of the neighbor discovery, the DHT lookup algorithm, and the stabilization of the virtual overlay depend on the specifications of the DHT. For our evaluation, we utilize a recursive Kademlia [22].

We have now presented the essential components of our design. In the following, we evaluate our design with regard to our requirements.

VI. Theoretical Analysis

The purpose of this section is twofold: First, we show that we indeed achieve sender and receiver anonymity in the form of plausible deniability. Second, we derive bounds on the routing and stabilization complexity in order to show that both indeed scale logarithmically or polylog with the network size n. We only sketch the proofs here. The complete proofs are contained in our technical report [21].

A. Anonymity

We show that our return addresses provide plausible deniability under the assumption that the attacker does not control the complete neighborhood of a sender or receiver.

Theorem VI.1. *Let u be a local attacker, which is aware only of its direct neighbors N_u in the social graph. Consider a request addresses for the return addresses $y = (y_1, \ldots, y_\gamma)$ with routing information $\tilde{k} = (\tilde{k}_1, \ldots, \tilde{k}_\gamma)$ for γ embeddings. Let A be a polynomial-time algorithm executed by u for identifying either the sender or receiver v of the request. Then u cannot identify v with absolute certainty, i.e., $P(A(y, \tilde{k}) = v) < 1$.*

So, we guarantee plausible deniability of both sender and receiver. The idea of the proof is to consider different cases with regard to the common prefix length of the receiver coordinate and attacker's closest neighbor. For each such case, we either show that none of the neighbors is the receiver or it is uncertain if the neighbor or one of its descendants is the receiver.

B. Scalability

In the first part of this section, we obtain upper bounds on the expected routing length of the routing algorithms $\mathbf{R}^{TD}$ and $\mathbf{R}^{CPL}$ for one embedding. The desired upper bound on the routing complexity follows by multiplying this bound for routing in one tree with τ, the number of trees used for parallel routing. Afterwards, we consider the stabilization complexity $CS^{\mathbf{S}}$ of the stabilization algorithm $\mathbf{S}$ consisting of i) the local reconstruction of the trees and ii) the assignment of new coordinates for the nodes affected by a change topology using the modified PIE embedding.

Theorem VI.2. *Let id be a modified PIE embedding on a spanning tree of G generated according to Section V-A with parameters γ and q. Furthermore, assume that the diameter of G is $diam(G) = \mathcal{O}(\log n)$. The expected routing length is at most $\mathbb{E}(R^{TD}) = \mathcal{O}\left(\frac{\gamma}{q}\log n\right)$ for the routing algorithm $\mathbf{R}^{TD}$ and $\mathbb{E}(R^{CPL}) = \mathcal{O}\left(\left(\frac{\gamma}{q}\right)^2 \log n\right)$ for $\mathbf{R}^{CPL}$.*

For the proof, we first show that the expected level of a node in the tree is bound by $\mathcal{O}\left(\frac{\gamma}{q}\log n\right)$ and then derive the routing length based on the expected level.

The bounds for a virtual overlay lookup follow directly from the fact that a DHT lookup requires $\mathcal{O}(\log n)$ overlay hops with each hop corresponding to one route in the network embedding, i.e., we multiple all the asymptotic bounds in Theorem VI.2 with $\mathcal{O}(\log n)$. We now derive the stabilization complexity for the spanning trees, the stabilization complexity of the virtual overlay depends on the overlay structure.

Theorem VI.3. *We assume the social graph G to be of a logarithmic diameter and a constant average degree. Furthermore, we assume the use of a the root election protocol with complexity $\mathcal{O}(n \log n)$. Then the expected stabilization complexity $CS^{\mathbf{S}}$ of the spanning trees for one topology change is $\mathbb{E}(CS^{\mathbf{S}}) = \mathcal{O}\left(\gamma\frac{\gamma}{q}\log n\right)$.*

We show that the expected stabilization complexity for one embedding is equal to the expected level of a node. The result then follows from the proof of the routing complexity.

We have shown that the complexity of routing, content discovery, and stabilization is bound (poly-)log as required.

VII. Simulations

In this section, we evaluate the efficiency and the resilience of our scheme in a simulation study. We start by validating the asymptotic bounds derived in Section VI-B for concrete scenarios. Afterwards, we consider the robustness to failures and the resistance to attacks.

A. Efficiency

We start by detailing our simulation model and set-up, followed by our expectation, the results and their interpretation.

Model and Evaluation Metrics : In order to evaluate the efficiency, we consider the routing length and the stabilization complexity. We express the stabilization complexity in terms of the average number of coordinates that have to reassigned when a randomly chosen node leaves, i.e., the average number of descendants of a node.

We compared our results to those for Freenet, a virtual overlay VO, and the original PIE embedding. The virtual overlay VO combines the advantages of X-Vine [11] and MCON [3] by using shortest paths as tunnels in a Kademlia overlay like MCON but integrating backtracking in the presence of local optima and shortcuts from one tunnel to another like X-Vine.

Set-up : Due to space constraints, we restrict the presented results to one example network, namely the giant component of a community network from Facebook with 63392 users [3].

The spanning tree construction is parametrized by the number of trees $\gamma \in \{1, 2, 3, 5, 7, 10, 12, 15\}$, the acceptance probability $q = 0.5$, and the parent selection criterion W chosen to be either random selection (denoted *DIV-RAND*) or preference of nodes at a low depth (denoted *DIV-DEP*). In addition, we consider a breadth first search for spanning tree construction (denoted *BFS*). Moreover, we consider the impact of the two distances δ_{TD} (denoted *TD*) and δ_{CPL} (denoted *CPL*). The length of the return addresses was set to $L = 128$ and the number of bits per element was $b = 128$, all $\tau = \gamma$ embeddings were considered for routing.

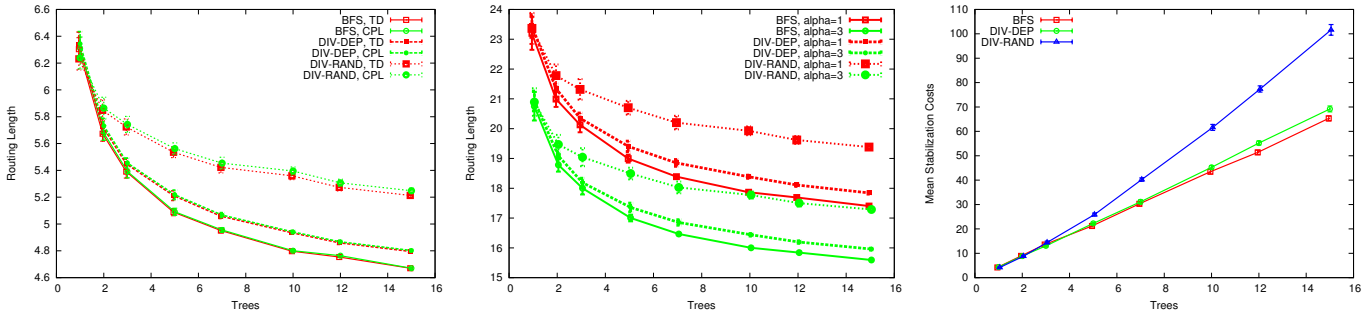
For the virtual overlay used for content addressing, we chose a highly resilient recursive Kademlia [22] with bucket size 8 and $\alpha \in \{1, 3\}$ parallel look-ups. Because routing table entries are not uniquely determined by Kademlia identifiers, the entries were chosen randomly from all suitable candidates.

We parametrized the related approaches as follows. For simulating Freenet, we executed the embedding for $6,000$ iteration as suggested in [23] and then routed using a distance-directed depth-first search based only on the information about direct neighbors. The routing and stabilization complexity of the original PIE embedding is equal to the respective quantities of our algorithm for $\gamma = 1$, the distance function δ_{TD} and routing without the use of backtracking. In order to better understand the results of the comparison, we simulate the virtual overlay VO using the same Kademlia overlay as for our own approach but replacing the tree routing by tunnels corresponding to the shortest paths between overlay neighbors. So, we parametrized the related approaches by either using the proposed standard parameters or selecting parameters that are suitable for comparison because they corresponds to the same degree of redundancy as the parametrization of our own approach.

All results were averaged over 20 runs. They are displayed with 95% confidence intervals. Each run considered $100,000$ randomly selected source-destination pairs.

Results : The impact of the three parameters, number of trees, tree construction, and distance on the routing length

[3] http://konect.uni-koblenz.de/networks/facebook-wosn-links



(a) $\mathbf{R}_{node}$ (b) $\mathbf{R}_{content}$ (c) Stabilization

Fig. 1: Impact of number of embeddings γ, tree construction, and distance function on routing length for a) tree routing and b) Kademlia lookup with degree of parallelism α; related approaches result in routing lengths of 14 (virtual overlay VO) and close to $10,000$ (Freenet), and c) stabilization overhead

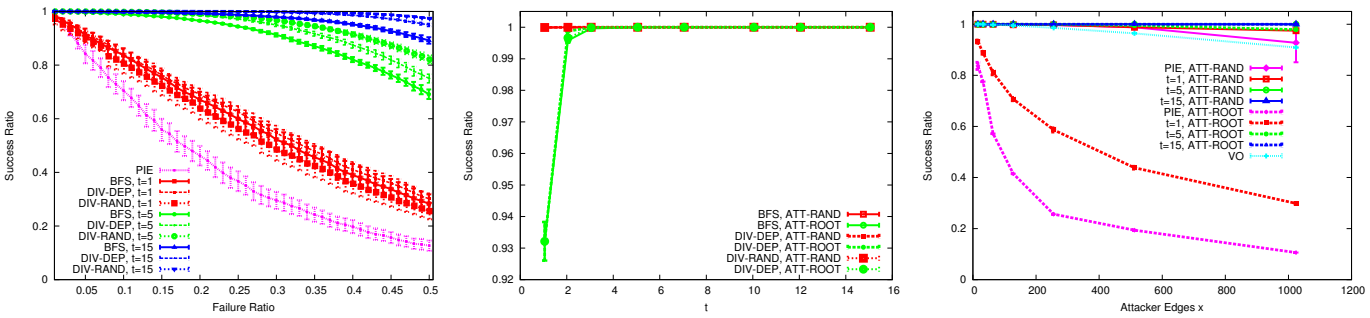
was in accordance with our expectations when designing the system. First, the results indicate that the tree construction, in particular the number of trees, is the dominating factor for the routing length. So, the routing length decreased considerably if multiple embeddings were used because the shortest route in any of the trees was considered. Second, preferring parents closer to the root, i.e., using *BFS* or *DIV-DEP*, produced shorter routes in the tree and hence reduced the routing length. Third, in comparison to the tree construction, the choice of a distance function had less impact. For *BFS* or *DIV-DEP*, the advantage of *TD* over *CPL* was barely noticeable, whereas the difference for *DIV-RAND* was still small but noticeable. In order to understand this difference, note that *CPL* is expected to lead to longer routes. The reason for the longer routes lies in forwarding the request to neighbors at a higher depth, which might have a long common prefix but are nevertheless at a higher distance from the destination due to their depth. For *BFS* or *DIV-DEP*, the difference of the depth of neighbors was generally small because neighbors at a lower depth were preferably selected as parents. In contrast, *DIV-DEP* allows for larger differences in depth. Hence there is a higher probability to increase the tree distance by selecting a neighbor with a longer common prefix length but at a high depth. All in all, the routing length varied between 4.67 (*BFS*, $\gamma = 15$, *TD*) and 6.24 (*DIV-RAND*, $\gamma = 1$, *CPL*) hops, as displayed in Figure 1a. In summary, the use of multiple embeddings indeed reduced the routing length considerably.

The performance of the DHT lookup in the virtual overlay directly related to the previous results (cmp. Fig. 1b for the distance under *TD*). The overhead for the discovery of a randomly chosen Kademlia ID, stored at the node with the closest ID in the overlay, varied between 15.56 and 24.25 hops in the F2F overlay, at around 4 hops in the virtual overlay.

By Theorem VI.3, the stabilization complexity was expected to increase at most quadratic with the number of trees. Indeed, Figure 1c supports this fact for *DIV-RAND*. The increase for *BFS* and *DIV-DEP* was even only linear and slightly super-linear, respectively. Note that the quadratic increase is due to the raising average depth of additional trees. With the goal of achieving diverse spanning trees, nodes select parents at a higher depth. However, the average number of descendants increases with the depth, because a node at depth l is a descendant of l nodes. Due to the stabilization complexity corresponding to the number of the departing node's descendants, the stabilization overhead was higher for *DIV-RAND* and *DIV-DEP* than for *BFS*. More precisely, *BFS* constructs all γ trees independently, so that the average depth of each tree is independent of the number of trees. The stabilization complexity per tree thus remains constant. *DIV-DEP*, aiming to balance diversity and short routes, causes stabilization overhead between the two former approaches, but performed closer to *BFS* (this similarity also held for the routing length). More concretely, the average stabilization overhead for a departing node was slightly below 4.5 for a single tree. For $\gamma = 15$, the overhead increased to 65 (*BFS*), 69 (*DIV-DEP*), and more than 101 (*DIV-RAND*). In contrast to a complete re-computation of the embedding requiring at least $n = 63392$ messages, the stabilization overhead is negligible.

For the related approaches, we found a routing length of 9403.1 for Freenet, 16.11 for VO with $\alpha = 1$, and 14.07 for VO with $\alpha = 3$. Furthermore, the shortest paths are on average of length 4.31, meaning that our routing length of 4.67 is close to optimal. So, routing between nodes in the tree required less than half the overhead of state-of-the-art approaches. Routing in the virtual overlay, requiring at best less than 16 hops in our scheme, was slightly more costly in our approach than in VO due to the inability of the tree routing to guarantee shortest paths between virtual neighbors.

A straight-forward comparison of the stabilization overhead was not possible. Since Freenet stabilizes periodically, there is no overhead directly associated with a leaving node. In case of virtual overlays, VO uses flooding for stabilization, which is clearly more costly. Other overlays such as X-Vine use less costly stabilization but stabilization and routing overhead are unstable and increase over time as shown in [4], so that it is unclear which state of the system should be considered for a comparison. In order to nevertheless give a lower bound on the stabilization overhead, we computed the number of tunnels that needed to be rebuild in VO. On average, 477.35 tunnels corresponding to shortest paths were affected by a



(a) Robustness (b) Attacks: 16 Edges (c) Attacks: Up to 1024 edges

Fig. 2: a) Robustness to failures for distance *CPL* and b),c) Censorship-Resistance of tree routing for distance *CPL* to adversaries which are either able to undermine the root election (*ATT-ROOT*) or are unable to do so (*ATT-RAND*) for b) $x = 16$ attacking edges, and c) up to $1,024$ attacking edges and tree construction *DIV-DEP*

departing node. If a tunnel is repaired by routing in the Kademlia overlay like in X-Vine, the stabilization overhead per tunnel corresponds to routing a request and the corresponding reply, i.e., for tunnels corresponding to shortest paths at least $2 \cdot 14 = 28$ messages, resulting in a lower bound on more than $10,000$ messages per node departure.

B. Resilience

Before detailing our simulation in detail, we derive attacker strategies for a local adversary unaware of the topology of the social graph. Here, we only state the strategy. Our technical report goes into more detail with regard to their applicability and explain why alternative attack strategies are bound to result in less damage [21].

Attack Strategies : In order to model secure and insecure root selection protocols, we consider two realizations of *ATT-RAND* and *ATT-ROOT*. In the following, assume that one attacker node has established x links to honest nodes and now aims to censor communication.

For secure spanning trees, the adversary A is unable to manipulate the root election. Nevertheless, A can manipulate the subsequent embedding. The attack strategy *ATT-RAND* assigns each of its children a different random prefix rather than the correct prefix. In this manner, routing fails because nodes in the higher levels of the tree do not recognize the prefix. So, the impact of the attack is increased in comparison to a random failure.

In contrast, if the adversary A can manipulate the root election protocol, *ATT-ROOT* manipulates the root election in all spanning trees such that A becomes the root in all trees. Under the assumption that the root observes the maximal number of requests, the attack should result in a high ratio of failed requests.

Simulation Model and Metrics : We utilized the simulation model and set-up from Section VII-A for evaluating the efficiency and extended it to include robustness and censorship-resistance. So, we simulate the robustness of an overlay by subsequently selecting random failed nodes. In each step, we select a certain fraction of additional failed nodes and then determine the success ratio. Furthermore, we evaluate attacks using the two attack strategies *ATT-RAND* and *ATT-ROOT* described above. We compare our results to the virtual overlay VO, described in Section VII-A, with an attacker that dropped all messages but did not strategically manipulate the protocols.

Set-up : We used the embedding and routing algorithms as parametrized in Section VII-A. In order to evaluate the robustness, we removed up to 50% of the nodes in steps of 1%. During the process of removing nodes, individual nodes inevitably became disconnected from the giant component, so that routing between some pairs was no longer possible. For this reason, we only considered the results for source-destination pairs in the same component. Our results are presented for 1, 5, and 15 trees only.

The number of edges x controlled by the adversary A were chosen as $x = 2^i \times \lceil \log_2 n \rceil$ with $0 \leq i \leq 6$ and $\lceil \log_2 n \rceil = 16$. So, up to $1,024$ attacker edges were considered. In particular, $x = 1024 > \frac{\sqrt{n}}{\log n}$, a common asymptotic bound on the number of edges to honest nodes considered for Sybil detection schemes [17]. For quantifying the achieved improvement, we compared our approach to the resilience of the original PIE embedding and routing, i.e., 1 tree, δ_{TD}, and no backtracking. For VO, we used a degree of parallelism of $\alpha = 1$. Since backtracking was applied, all values of $\alpha > 0$ resulted in the same success ratio, because regardless of the value of α, the routing succeeded if and only if a path along which the distance of virtual neighbors decreases in each hop.

Results : First note that the observed differences between the two distances δ_{TD} and δ_{CPL} were negligible, i.e., less than 0.1%. *TD* had a slight advantage for random failures due the routes and the resulting lower probability in encountering a failed node. In contrast, *CPL* entailed a higher resistance to attacks because of the lower importance of strategically important positions. However, as the differences were barely noticeable, we present the results for *CPL* in the following with the exception of the results for the original PIE embedding.

We start by evaluating the robustness to random failures. The results, displayed in Figure 2a, indicate that the use of multiple embeddings considerably increased the robustness. The success ratio for $\gamma = 1$ was low, decreasing in a

linear fashion to less than 30% for a failure ratio of 50%. In contrast, for $\gamma = 15$, the success ratio exceeded 90%. Though the number of embeddings was the dominating factor, the tree constructing algorithm also strongly influenced the success ratio. For $\gamma > 1$, aiming to choose distinct parents improved the robustness to failures because of the higher number of distinct routes. For example, when routing in 5 parallel embeddings, the success ratio was above 80% for *DIV-RAND*. In contrast, *BFS* had a success ratio below 70%. In summary, the robustness to failures was extremely high for multiple embeddings, enabling a success ratio of more than 95% for up to 20% failed nodes. The robustness was further increased by using *DIV-RAND* or *DIV-DEP* rather than *BFS*, showing that even such relatively simple schemes can achieve a noticeable improvement.

Now, we consider the censorship-resistance for $x = 16$ attacking edges, as displayed in Figure 2b. If the adversary A was unable to manipulate the root selection, the success ratio was only slightly below 100%. Even if $\gamma = 1$, more than 99.5% of the routes were successfully discovered. The high resilience against *ATT-RAND* was to be expected, considering that the attack was only slightly more severe than failure of one random node. If the attacker was able to become the root in all trees, the success ratio dropped to about 93% for $\gamma = 1$. However, with multiple trees, the ratio of *ATT-ROOT* was close to 100%. The impact of the tree construction was small but noticeable. So, *BFS* generally resulted in a slightly lower success ratio. Hence, by using multiple embeddings and backtracking, the resilience to an adversary that can establish only $\lceil \log_2 |V| \rceil = 16$ is such that nearly all routes are successfully found.

For an increased number of attacking edges x, the success ratio remained close to 100% when more than one tree was used for routing, as displayed in Figure 2c for *DIV-DEP*. However, for one tree, the success ratio decreased drastically if an attacker could undermine the root selection. For $x = 1024$, i.e., if the attacker controlled edges to roughly 1.7% of the nodes, the success ratio for $\gamma = 1$ decreased to slightly less than 30%. In contrast, if $\gamma = 5$ or $\gamma = 15$, the success ratio was still 97.9 or 99.9%, respectively.

We compared the results for our approach with the original PIE embedding and the virtual overlay VO. As can be seen from Figure 2a, the success ratio dropped much more quickly for PIE than for the improved approaches. In contrast to PIE, VO exhibited a rather high success ratio as displayed in Figure 2c. VO's advantage in contrast to $\gamma = 1$ holds despite VO's longer routes (see Section VII-A). The reason for VO's lower vulnerability lies in the absence of strategic manipulation. While greedy embeddings allow the attacker to assume an important role, our attacker in VO does not attract a disproportional fraction of traffic. However, establishing multiple trees ensures that the role of the root is effectively mitigated, so that the censorship-resilience of VO is slightly lower than VOUTE's resilience for 5 or more parallel embeddings.

All in all, the simulation study confirms that our F2F overlay design exceeds the related approaches with regard to both efficiency and resilience, with the exception of the DHT routing.

VIII. Conclusion

We have introduced a privacy-preserving, efficient, and resilient design for F2F overlays. For this purpose, we have developed an algorithm for the generation of anonymous return addresses Furthermore, we have designed multiple parallel network embeddings to enable both efficiency and resilience, as validated by an extensive simulation study.

Extending our simulation results, we are currently integrating our algorithms in an existing F2F overlay and have started initial testbed studies to better understand the system and its performance in real environments.

Acknowledgements

The presented work was partially founded by the DFG (Grant STR 1131/2-1), BMBF project SEnCom, the CRC HAEC, and the Excellence Cluster "cfaed". Furthermore, the authors want to thank Andreas Höfer for helpful discussions.

References

[1] R. Dingledine et al. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.
[2] N. Evans et al. Routing in the dark: Pitch black. In *ACSAC*, 2007.
[3] E. Vasserman et al. Membership-concealing overlay networks. In *CCS*, 2009.
[4] S. Roos and T. Strufe. On the impossibility of efficient self-stabilization in virtual overlays with churn. In *INFOCOM*. IEEE, 2015.
[5] R. Kleinberg. Geographic routing using hyperbolic space. In *INFOCOM*, 2007.
[6] J. Herzen et al. Scalable routing easy as pie: A practical isometric embedding protocol. In *ICNP*, 2011.
[7] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy*, 2009.
[8] B. Popescu. Safe and private data sharing with turtle: friends team-up and beat the system. In *Security Protocols*, 2006.
[9] T. Isdal et al. Privacy-preserving p2p data sharing with oneswarm. In *ACM SIGCOMM CCR*, 2010.
[10] N. Evans and C. Grothoff. R5n: Randomized recursive routing for restricted-route networks. In *NSS*, 2011.
[11] P. Mittal et al. X-vine: Secure and pseudonymous routing in dhts using social networks. In *NDSS*, 2012.
[12] A. Hofer et al. Greedy embedding, routing and content addressing for darknets. In *NetSys*, 2013.
[13] CH Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. In *ALGOSENSORS*. 2004.
[14] A. Cvetkovski and M. Crovella. Hyperbolic embedding and routing for dynamic graphs. In *INFOCOM*, 2009.
[15] D. Eppstein and M. Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In *Graph Drawing*, 2009.
[16] H. Mohajeri Moghaddam et al. Skypemorph: Protocol obfuscation for tor bridges. In *CCS*, 2012.
[17] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.
[18] A. Singh et al. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM*, 2006.
[19] J. Liang et al. The index poisoning attack in p2p file sharing systems. In *INFOCOM*, 2006.
[20] R. Perlman. An algorithm for distributed computation of a spanningtree in an extended lan. *ACM SIGCOMM CCR*, 15(4), 1985.
[21] S. Roos et al. Voute-virtual overlays using tree embeddings. *CoRR*, abs/1601.06119, 2016.
[22] B. Heep. R/kademlia: Recursive and topology-aware overlay routing. In *ATNAC*, 2010.
[23] O. Sandberg. Distributed routing in small-world networks. In *ALENEX*, 2006.