End-to-End Key Exchange through Disjoint Paths in P2P Networks

Daouda Ahmat, Damien Magoni, Tegawendé F. Bissyandé

ICST Transactions Preprint

End-to-End Key Exchange through Disjoint Paths in P2P Networks

Daouda Ahmat[1], Damien Magoni[1,*], Tegawendé F. Bissyandé [2]

[1]LaBRI, University of Bordeaux, France
[2]SnT, University of Luxembourg, Luxembourg

Abstract

Due to their inherent features, P2P networks have proven to be effective in the exchange of data between autonomous peers. Unfortunately, these networks are subject to various security threats that cannot be addressed readily since traditional security infrastructures, which are centralized, cannot be applied to them. Furthermore, communication reliability across the Internet is threatened by various attacks, including usurpation of identity, eavesdropping or traffic modification. Thus, in order to overcome these security issues and allow peers to securely exchange data, we propose a new key management scheme over P2P networks. Our approach introduces a new method that enables a secret key exchange through disjoint paths in the absence of a trusted central coordination point which would be required in traditional centralized security systems.

Keywords: P2P networks, key management, Diffie-Hellman algorithm, MITM attacks, multipath routing, backtracking.

1. Introduction

Due to their inherent characteristics, including self-organization, availability, reliability, scalability, and their potential for managing load balancing and dynamic topology changes, P2P networks remain one of the prime choices to provide affordable means for sharing data, publishing information and streaming media.

Distributed systems, and especially P2P networks, can effectively operate in dynamic environments because they are free from relying on a single point of failure that a central infrastructure would represent. However, although these systems provide interesting properties, they are not suitable for integrating approved traditional centralized security infrastructures. Therefore, in the context of P2P system setups, security remains challenging to implement and maintain.

Without guarantee of provision of a central infrastructure, P2P networks are bound to face security and reliability issues that existing common policies and techniques fail to take into account in their implementations. For example, standard security measures for communications involve the use of key-based encryption which is often implemented with public-key cryptography. However, a central problem with the use of such type of cryptography lies in confidence that a given public key is authentic, this means that it is correct and belongs to the entity claimed, that it has not been tampered with or that is has not been replaced by malicious third party. Usually, this confidence is guaranteed by a central Public-Key Infrastructure (PKI), in which one or several certification authorities certify ownership of the public keys. There is therefore a requirement for a strong centralization to manage cryptographic keys, a luxury that P2P networks cannot practically afford [26].

In order to overcome the limitations imposed by the incompatibility between traditional security policies and decentralized P2P networks, we propose in this paper a new approach for key management in the absence of a centralized infrastructure. Our approach takes into account the specific features of P2P networks and the security and reliability requirements for exchanging information in this area. In opposition to the traditional PKI, the key management in our scheme is *decentralized*. Indeed, this management is distributed among both nodes of a communicating pair of nodes, meaning that the key negotiation model is based upon an End-to-End (E2E) exchange scenario. In practice, we rely on a scheme that uses multiple paths (hereafter referred to as *multipath*) for performing a Diffie-Hellman (DH) key exchange where each source node can select several disjoint paths inside the P2P network in order to prevent Man-In-The-Middle (MITM) attacks. Accordingly, key parts (hereafter referred to as *subkeys*) can then be forwarded safely through these disjoint paths to ensure communication security.

*Corresponding author. Email: magoni@labri.fr

In our related work [2], we have proposed an approach to address the security issues of communication sessions when users are mobile across networks. As this approach, which is targeting user-level applications, is based on a P2P network, it can leverage the solution proposed in this paper. Indeed, this solution solves the security issues arising from the intermediate P2P network nodes and provides a PKI-less solution to securely create a secret key whose distribution was undefined in [2]. This paper is an extended version of our previous work [1] which was presented at the 5th EAI International Conference on e-Infrastructure and e-Services for Developing Countries in 2013. We have increased the description of our solution and detailed its implementation and we have added simulation results showing that our solution can secure a communication with high probability depending on the network conditions.

The main contributions of this paper are:

- We discuss the challenges that arise in P2P networks, focusing on the safety and reliability of communications. We then discuss the opportunities of multipath key exchange for P2P networks: how they can be harnessed to deliver secure communication in a truly beneficial way. We expose the challenges for implementing the authentication of source nodes in a P2P network. We emphasize on why traditional PKI, which are currently successful on the Internet, are inadequate for P2P systems (Section 2).
- We describe the design of our solution in detail, including the discovery of multiple paths and the use of backtracking (Section 3).
- We present a security analysis of our solution, which is based on the probability of having a given set of paths intercepted by malicious nodes (Section 6).
- We evaluate the efficiency of our solution by carrying out experiments by simulations and we provide detailed results for assessing its usefulness (Section 5).

The remainder of this paper is organized as follows. Section 2 presents background information including the challenges that must be overcome to secure communications in P2P networks as well as some insights from our previous work on P2P overlay networks including our implementation of mobile secure sessions. We present the design of our approach in Section 3. Section 6 provides an analysis about the weaknesses and strengths of our scheme. Experiments carried out by simulation are detailed in Section 4 and the results are outlined in Section 5. Finally, we discuss the related work in Section 7.

2. Background

Securing communications in P2P networks is a challenging endeavor, especially with regards to the standard practice of encrypting information with the assurance that receivers have knowledge of the public key for deciphering the data payload. In this section, we precisely detail some obvious and non-obvious challenges to highlight the constraints of finding a solution to key management for P2P systems. We then introduce the concept of Hyperbolic coordinates which are leveraged in our approach.

2.1. Challenges

- *Key distribution.* The first challenge that we encounter is the mode of distributing cryptographic keys in P2P networks. Indeed, in traditional networks, a management-friendly central infrastructure, called PKI, is relied upon for this task. In the absence of such infrastructure for P2P networks, it is important to devise a fully distributed approach to spread keys reliably. This approach should take into account the volatility of P2P networks but could leverage their self-organization property.
- *Assurance of alternate infrastructure.* Relying on the P2P networks to assure the forwarding of cryptographic keys also comes with problems that a traditional PKI was able to easily handled. Indeed, there is a new need to ensure in a distributed system that the construction of the overlay network where each peer is properly identified will guarantee the robustness of the exchange scenario with little possibility of corruption by any intermediate peer.
- *Exchange of keys.* When initiating a communication in a P2P network, there will be a need for the peers to agree on the generation of a session key that the two peers will shared. A challenge in this requirement lies in the negotiation which should also be secured. Although usage of cryptography asymmetric algorithm to secure information is expensive, it can be leveraged in the initial step that is the key negotiation phase.
- *Detection of attacks.* Last but not least, we note the issue of providing a mechanism for detecting corrupted keys. Indeed, when a corrupted cryptographic key is introduced, it should be detected and then revoked by peers who are aware of the corruption. This information should be included in a notification message that must be sent by broadcast through the network. Thus, the system will assure that all corrupted keys are flushed out of the memory of connected peers. Similarly, a

challenging endeavor will be to prevent MITM attacks during session key negotiation phase

2.2. Hyperbolic Coordinates

In order to address the security challenges of P2P networks, we propose a solution that will be used on top of the CLOAK [1] P2P architecture defined in our previous work [25]. The CLOAK architecture is appealing because it provides an addressing scheme based on virtual hyperbolic coordinates and a greedy routing based on the hyperbolic distance between nodes computed by using those coordinates. Thus CLOAK does not use routing tables and does not impose any specific type of topology upon the nodes of the network.

We have demonstrated in [5] that the hyperbolic geometry can be efficiently used for addressing and routing in P2P overlay networks. Indeed, the hyperbolic plane provides a means for distributing unique coordinates taken from an infinite q-regular tree (with q being an integer chosen at will). These coordinates are used as addresses by the network nodes. When the P2P network expands, starting from a root node, any joining node will ask for an address from one of the existing nodes. The joining of new nodes thus create an *addressing* spanning tree. Upon obtaining an address, a newly included node will then be able to independently compute the addresses that it will be able to provide to its potential children and so on. The degree q of the regular tree determines how many addresses each node in the network will be able to give to future joining nodes.

3. System Design

In this section, we describe our key exchange model and its corollaries. We present the design of the variety of features that are in play for realizing the goals of a robust key exchange. These include the multipath routing approach, a backtracking algorithm, an authentication mechanism, the use of proxies in case of negotiation failure, and the addition of extra properties to the generated keys.

3.1. Approach Overview

In recent years, a number of research efforts have emphasized that P2P networks are subject to various security threats due to their inherent features [6, 9, 18, 21, 22]. To overcome the identified security issues in P2P networks, we propose to design a new key exchange scheme that provides end-to-end (E2E) authentication and confidentiality capabilities. The proposed key exchange protocol is an extension of the well known DH cryptographic algorithm. Our approach mainly aims at preventing MITM attacks by using multipath key exchange technique.

In our approach, the main key which must be transferred between the source and destination nodes is split into several parts called subkeys. Each subkey is sent to the destination node through separate paths over the P2P network. Thus, a malicious node eavesdropping on a particular overlay link will only recover one subkey but would not be be able to recover all other subkeys. Therefore, the main key, which consists of all subkeys, will remain unknown to the malicious node. Similarly, in the case of a coordinated attack involving many nodes eavesdropping on different links, they must be able to intercept all subkeys for their attack to be successful as all subkeys are needed to reconstruct the main key. An incomplete subkey collection will not allow any node (intended recipient or attacker) to infer the main key.

In order to fulfill the goal of diversifying the links used to transfer the subkeys, we propose a routing algorithm based on the use of multipath to route each subkey separately. The marking algorithm is an essential subroutine in our routing algorithm to route split key parts via disjoint paths between the source and destination nodes. Thus, a multipath key exchange scenario could prevent MITM attacks and ensure confidential communication. We further complete our architecture with features such as the designation of *trusted peers* and backtracking capabilities.

3.2. Multipath Routing and Backtracking Algorithm

Goals. For reasons of simplicity, in the remainder of this paper, we represent a P2P network by means of an undirected graph $G = (V, E)$, where V is a non-empty set of vertices and E the set of edges between pairs of vertices. Given two distinct nodes $\{s, t\} \in V$, we want to find k disjoint paths $P_0, ..., P_{k-1}$ over G from s to t. We also assume that each node knows only its immediate neighbors and has no knowledge of the global topology of the graph.

Each P_i is a collection of n selected consecutive hops $h_{i_0}, ..., h_{i_{n-1}}$. Each hop h_{i_j} in path P_i represents an optimal hop at step j that meets routing protocol criteria. A path P_i is formally defined as follow:

$$P_i = \{\langle h_{i_0}, ..., h_{i_{n-1}}\rangle_{0\le i\le k-1} \mid for\ 0 \le j \ne l \le k-1, \\ and\ for\ 0 \le p, q \le n-1,\ h_{j_p} \ne h_{l_q}\} \quad (1)$$

The main key is composed of a set K of subkeys, of size ω. Formally, let $K' \subset K$ such that $|K'| < \omega$, then the combination of all subkeys $\in K'$ cannot determine the main key. Thus, at the end of the transmissions, to recompute the main key, all ω subkeys are needed.

[1] Covering Layers Of Abstract Knowledge

This property is needed in our security infrastructure to introduce a robust key exchange policy.

Finding k Disjoint Paths. In our context, the *disjoint paths* term means vertex/node-disjoint paths. Two paths are vertex-disjoint if they have no vertex in common except for the first (source) and last (destination) vertices. In order to find k disjoint paths over graph G, we proceed by relying on a graph marking algorithm that enables to mark all hops which make up a separate path P that fulfills properties presented by equation (1). Concretely, information about each visited node is stored into a *visited list* (V) contained within the subkey transfer packet. Thus, step by step, all visited node will be recorded into V. On receipt of a subkey packet, the destination node replies to the source node through an acknowledgement (ACK) message that contains among other information an updated list V and the sequence number of the received packet. This process is iterated k times for the requested k separate paths. In the ideal case described by equation (2), all used paths between the source and destination nodes are fully pairwise disjoint.

$$P_i \cap P_{j\,0\leq i\neq j\leq n-1} = \emptyset \tag{2}$$

Furthermore, although some paths may intersect at some nodes, if these paths verify equation (3), MITM attacks could be prevented. Unfortunately, in this scenario, key exchange would be more vulnerable to coordinated MITM attacks launched by malicious nodes placed on the intersection points.

$$\bigcap_{i=0}^{k} P_i = \emptyset \tag{3}$$

A node N is qualified to become a hop if, and only if, $N \notin V$ and N is an *optimal choice* at current step to reach the destination node.

In order to avoid redundancy, we assume that all paths P_i do not contain source and destination peers because these nodes are already recorded in the packet to indicate the origin and the final recipient of the subkey being transmitted.

Options for Routing Policies. Several multipath routing algorithms can be used to exchange security keys over P2P networks. However, these routing algorithms have different robustness and complexity characteristics. We now describe some common routing methods to highlight various choice-impacting criteria.

1. *combining routing and marking.* A first routing algorithm enables to perform both marking and routing at the same time. Key components are successively sent from source to destination through nodes that are marked as soon as they are used as hops (not *just visited* nodes) to reach the destination node. Thus, each subkey packet is augmented with information of all hops that it goes through. This process is repeated for all the k subkeys. In the worst case, the running time of this algorithm is $\tau_1 = \mathcal{O}(k(|E| + |V|log|V|))$, where E represents the set of edges, V is the set of vertices and k represents the number of subkeys. The space complexity of this algorithm amounts to $\sigma_1 = \mathcal{O}(k(|V|))$. Unfortunately, this approach is unable to determine proportionality between the number of subkeys and the existence of enough disjoint paths. Indeed, the key is split before the process for finding disjoint paths is launched.

2. *pre-routing then routing.* This method requires two successive steps. In the first step, the source node tries to find disjoint paths between the source and destination nodes over the network by marking visited nodes. In the second step, refers to the obtained list of visited nodes to split the key and send them separately. The running time for this approach is $\tau_2 = |V| * \tau_1 = \mathcal{O}(k|V|(|E| + |V|log|V|))$. IN the worst case scenario, its space complexity amounts to $\sigma_2 = k|V| * \sigma_1 = \mathcal{O}(k^2(|V|^2))$.

   Despite such high complexity costs, this method has a major benefit: it enables to safely split the key in a number of subkeys that is in adequacy with the number of disjoint paths that exist in the network. Unfortunately, as the topology may change in the meantime, the considered paths may become invalid when the subkeys are finally forwarded.

3. *discovery and routing.* This method consists in a combination of the algorithms described above for creating a scheme where after each path discovery a subkey packet is sent. A packet discovery is indeed sent first to discover a path from source to destination. Then, a subkey is sent. This process is repeated in k times. The algorithm thus has the same complexity costs as the second method.

4. *reduction of the disjoint paths problem to a max-flow problem in undirected graphs.* The idea behind this approach is based on Menger's theorem[2] (1927) [4].

   The problem of finding disjoint paths can be reduced to a *Max-flow* finding problem. Double reduction is needed to achieve this goal: (1) reduction from vertex-disjoint paths problem to edge-disjoint paths problem and (2) reduction from edge-disjoint paths problem to *Max-flow* finding problem. This approach assigns unity

[2] "Let a finite undirected graph $\mathcal{G}$ and two distinct nodes $\mathcal{S}$ and $\mathcal{T}$, the maximum number of edge-disjoint $\mathcal{S}-\mathcal{T}$ paths is equal to the minimum number of edges whose removal disconnects $\mathcal{S}$ from $\mathcal{T}$."

capacity to every edge and then determines *Max-flow* between source($\mathcal{S}$) and destination($\mathcal{D}$) over the network. The *Max-flow* value computed is then precisely equal to the number of disjoint paths between $\mathcal{S}$ and $\mathcal{D}$. Although theoretically interesting, the approach is currently impractical, as it requires a global knowledge on network topology. This requirement is not fulfilled in P2P fully decentralized systems.

Recently, Ken-ichi *et al.* have proposed an algorithm that improves the time complexity of the disjoint paths problem by solving it in quadratic time, i.e., in $\mathcal{O}(n^2)$ [13].

Thanks to its robustness, the second algorithm emerges as the best choice, even though it is more costly than other solutions. Thus, our key exchange scheme leverages the *pre-routing then routing* method described above.

Routing Policy Illustration. Figure 1 illustrates in simplicity the case of two disjoint paths between two nodes, namely *1* and *9*. First, the source node sends a first packet (with *visited list* $V = \emptyset$) that determines P_1 consisting of hops *4* and *7*. These hops are added into V. Then, on receipt of the acknowledgement message which contains the updated list V, the source node sends a second message (with $V = \{4, 7\}$) that selects optimal hops without contacting any node already visited. Thus, P_2 will consist e.g. of nodes *3*, *6* and *8* (with $V = \{4, 7, 3, 6, 8\}$), such that $P_1 \cap P_2 = \emptyset$: two disjoint paths between source node *1* and destination node *2* have been found, and the two subkeys can be transferred.

Subsequently to receiving all subkeys, the destination node *9* can apply the symmetric operation of re-assembling the key in order to complete the key exchange process. At this time, and unless a coordinated attack have recovered all subkeys from all disjoint paths, both nodes *1* and *9* can start a secure communication using the exchanged key. The transmitted packets can even be routed through any nodes even those that were not included in V.

Description of our Routing Policy. When a source node $\mathcal{S}$ is attempting to exchange a key with a destination node $\mathcal{D}$, it splits this key into several subkey components that must be dispatched through separate channels[3]. We propose a straightforward technique to overcome the challenge of selecting the different paths that are necessary to route separately each subkey towards the destination node. As illustrated in the previous simple example, in the graph representing the nodes and connections in a P2P network, the degree of a node is the cardinal number of the set of neighbors which are known from start. Each connected node has a degree that may be less or equal to the network degree, which is the maximum number of neighbors.

[3] Algorithm 2 illustrates how the dispatching of key components is implemented over our infrastructure.

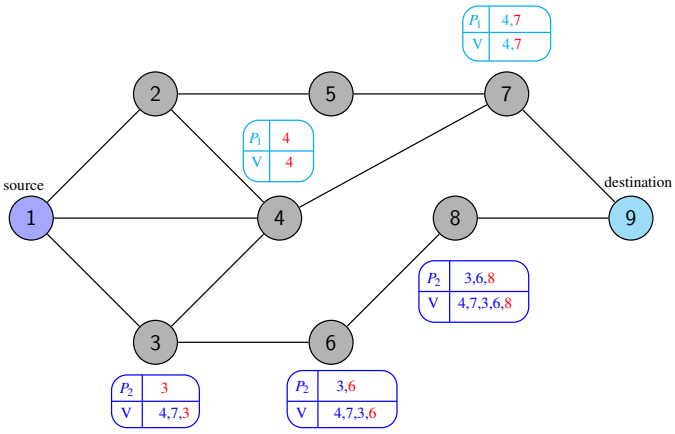


Figure 1. Routing across disjoint paths: the case of 2 disjoint paths.

Thus, given a network degree n, each node wishing to forward a key must split it into at most n components that will be routed in potential separate paths. The selection of paths is iterative to route each subkey through different neighbor. Thus, each node numbers the different nodes it is connected to and considers each of these nodes as the beginning of a different path. When a node has a degree k, it sends the first component to its first connected node, and the second component to the second connected node, and so on until all connected nodes are used if the number of disjoint paths is equal to the number of key components. We formally describe this path selection in Algorithm 1.

Algorithm 1: Neighbor selection and subkey routing.

```
Input: keyToTransmit, nodeNeighbors
usedNodesList ← ∅;
nodeNeighbors ← sortConnectedNodes(nodeNeighbors);
keyComponents ← splitKey(keyToTransmit);
node ← firstOf(nodeNeighbors);
while keyComponents ≠ ∅ do
    if node ∉ usedNodesList then
        component ← firstOf(keyComponents);
        transmitViaSeparatePath(component, node);
        keyComponents ← keyComponents \ {component};
        usedNodesList ← usedNodesList ∪ {node};
    node ← nextOf(nodeNeighbors, node);
```

Our solution can be used by any P2P network, whether unstructured or structured (such as DHT-based P2P networks). Indeed, the algorithm 1 is generic to any P2P network. However, the algorithm 2 is specific to our CLOAK P2P network.

Algorithm 2: Routing a subkey in a CLOAK network.

Input: CurrentPeer, SubKeyPacket, VisitedList
Output: Success
$w \leftarrow$ SubKeyPacket.$DestinationPeer.Coords$;
$m \leftarrow$ CurrentPeer.$Coords$;
$d_{min} \leftarrow \text{argcosh}\left(1 + 2\frac{|m-w|^2}{(1-|m|^2)(1-|w|^2)}\right)$;
$p_{min} \leftarrow$ CurrentPeer;
foreach $Neighbor \in p_{min}.Neighbors$ **do**
  IsAlreadyVisited $\leftarrow$ VisitedList.$contains$ ($Neighbor$);
  if IsAlreadyVisited = $false$ **then**
    $n \leftarrow Neighbor.Coords$;
    $d \leftarrow \text{argcosh}\left(1 + 2\frac{|n-w|^2}{(1-|n|^2)(1-|w|^2)}\right)$;
    if $d < d_{min}$ **then**
      $d_{min} \leftarrow d$;
      $p_{min} \leftarrow Neighbor$;
if $p_{min} \neq$ CurrentPeer **then**
  VisitedList.add (p_{min});
  routeSubKey (p_{min}, SubKeyPacket);
  if p_{min} = SubKeyPacket.$DestinationPeer$ **then**
    return $success$;
  else
    algorithm 2 (p_{min}, SubKeyPacket, VisitedList);
else
  backtracking();

Hop Number Threshold. In order to address the overheads in packets sizes, the number of hops is limited as illustrated in figure 2. Indeed, each packet used to transfer a subkey contains information on all nodes that are already visited in the current session, so as to allow the selection of paths that are will be disjoints. Thus, when the number of hops is significant then the portion of the packet that records the visited nodes grows, and can lead to an overflow in packet size. Consequently, it is necessary to limit the number of hops by a *threshold*.

Backtracking Algorithm. Backtracking algorithm is illustrated by a scenario depicted in Figure 3. First, path $\mathcal{P}_1 = \{1, 5, 8\}$ between $\mathcal{S}$ and $\mathcal{D}$ is determined. Then, an attempt to determine a second path $\mathcal{P}_2$ after performing hops on nodes *2*, *3* and *7* fails at node *7*, since continuing will not produce a disjoint path: nodes *5* and *8* is already used by $\mathcal{P}_1$ and there are no other ways from node *7*. As a result, a backtracking routine must be switched on. Discovery path packet will be sent back to node *3* and finally to node *2*, where it will attempt an alternate hop. Finally, the algorithm finds a second disjoint path $\mathcal{P}_2$ such that $\mathcal{P}_2 = 246$ ($\mathcal{P}_1 \bigcap \mathcal{P}_2 = \emptyset$). Note that all path

Formally, the backtracking algorithm enables to go back when the destination node is unreachable from current node or when the only path to reach it will intersect with previously selected paths. In such cases, the packet must sent backwards until either source node $\mathcal{S}$ is reached or until another relevant node for alternate paths is reached.

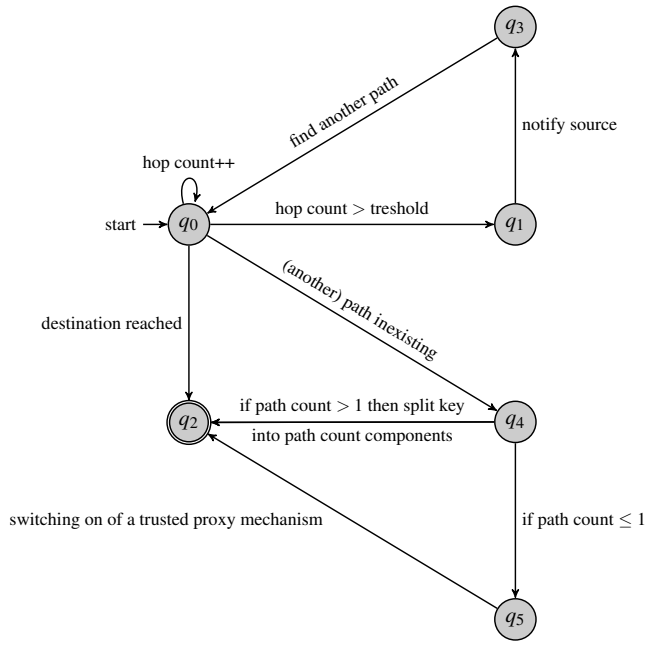


Figure 2. A threshold in the number of hops should prevent packet overhead; switching to a trusted proxy (see subsection 3.3) in unsuccessful cases.

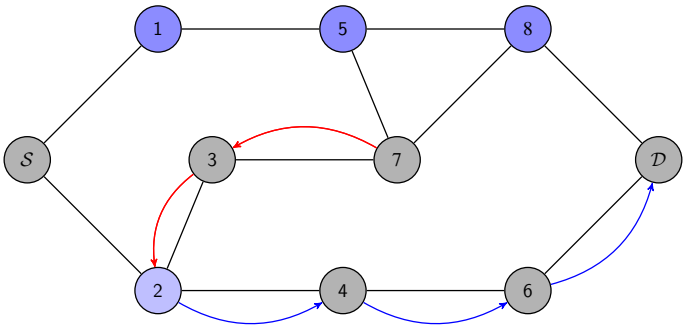


Figure 3. Backtracking illustration: there is no way out from node *7* to destination $\mathcal{D}$, because nodes *5* and *8* are marked.

3.3. E2E Key Exchange and Re-Authentication

E2E Key Exchange Scheme. In this section, we propose an extension that improves both the state-of-the-art models designed by Takano *et al.* [23] and Diffie & Hellman [8]. Scalable, decentralized and self-organized networks such as P2P systems enable many users to join them. Thus, each node can be connected to many other nodes; there can then be several paths between two endpoints of the network. For the above reasons, network topology can be transformed into a graph: each

Alice				sends each k_{a_i} / b_i via a disjoint path	Bob			
Data		*Actions*			*Actions*		*Data*	
secret	public	chooses randomly	computes		computes	chooses randomly	public	secret
s_a	p,g			$p, g \rightarrow$				s_b
...	p,g,A		$A = g^{s_a} (mod\ p)$				p,g	...
...	...	$k_{a_0}, ..., k_{a_{q-1}}$ such that	$\sum_{i=0}^{q-1} k_{a_i} (mod\ p) = A$	$k_{a_i}\ (0 \leq i < q) \rightarrow$			p,g,A	...
...	...				$B = g^{s_b} (mod\ p)$		p,g,A,B	...
...	p,g,A,B			$\leftarrow k_{b_i}\ (0 \leq i < q)$	$\sum_{i=0}^{q-1} k_{b_i} (mod\ p) = B$	$k_{b_0}, ..., k_{b_{q-1}}$ such that	...	...
s_a,**s**	...		$\mathbf{s} = B^{s_a} = g^{s_a s_b} (mod\ p)$	$\leftarrow$ Untrusted channels $\rightarrow$	$\mathbf{s} = A^{s_b} = g^{s_a s_b} (mod\ p)$		...	s_b,**s**

Figure 4. Data owned and actions performed by the source and destination nodes to negotiate the keys. Illustration of Algorithm 3.

node represents an edge of the graph and each link indicates a connection between two nodes.

Although the DH cryptographic algorithm has been widely used to share secrets on insecure communication channels, it is vulnerable to MITM attacks. In order to overcome this limitation of DH-based scenarios, we propose to use a multipath key exchange scheme. Building upon a scalable P2P overlay [5, 24] which uses virtual coordinates taken from the hyperbolic plane that is indifferent to underlying P2P network topology. Figure 4 summarizes the data and actions performed by a pair of nodes during a key negotiation phase. The multipath key negotiation is described more formally in Algorithm 3 and in figure 4. Takano *et al.* [23] have previously proposed a similar key negotiation mechanism. However, unlike our approach, their method is restricted only to Symphony and Chord P2P networks with a ring topology.

Trusted Peers Used as Proxies. Using *trusted peers* provides an alternative security solution when several disjoint paths between a source peer and a destination peer cannot be found. It is a key concept for addressing issues caused by the potential lack of disjoint paths in some network topologies. Thus, when a node fails to find disjoint paths through network, it activates the mechanism that enables to select one *trusted peer* from a *trusted peers* list and uses it as a proxy in order to securely reach the destination peer.

A node can be chosen from a list of nodes as *trusted peer* if it meets two conditions: (1) it is near to destination node than source peer except there is not another *trusted peer* and (2) it already exists a shared key between proxy node and source node. Thus, source peer will then delegate key negotiation process to its proxy node (see *request* message in figure 5).

Figure 5 presents a topology that shows that there is no disjoint paths between both nodes *1* and *2*. Therefore, using of a *trusted peer* (node *4* in this case) is then needed in this context.

trusted peer applies algorithm that enables to find disjoint paths (paths P_1 and P_2 use respectively hop *6* and hop *7* in figure 5) in order to exchange key with destination. Subsequently, Node *7* forwards key to source node by *reply* message sent over secure tunnel. Finally, node *1* and node *8* could then securely communicate without a *trusted proxy*.

Algorithm 3: Multipath key exchange scheme.

public parameters:
p : a prime number
g : a generator
secret parameters:
Alice : secret key s_a
Bob : secret key s_b

1. **Alice** selects a random number s_a and she then computes $Key_a = g^{s_a} (mod\ p)$;
2. **Alice** selects q random numbers $k_{a_0}, ..., k_{a_{q-1}}$, such that $Key_a = \sum_{k=0}^{q-1} k_{a_k} (mod\ p)$;
3. **Alice** routes all k_{a_k} to **Bob** through q potential separate channels ;
4. **Bob** receives q Key_a's components $k_{a_0}, ..., k_{a_{q-1}}$ sent by **Alice** and computes $Key_a = \sum_{k=0}^{q-1} k_{a_k} (mod\ p)$;
5. **Bob** selects a random number s_b and he then computes $Key_b = g^{s_b} (mod\ p)$;
6. **Bob** chooses q random numbers $k_{b_0}, ..., k_{b_{q-1}}$, such that $Key_b = \sum_{k=0}^{q-1} k_{b_k} (mod\ p)$;
7. **Bob** sends all k_{b_k} to **Alice** via q potential disjoint paths through network ;
8. **Alice** receives q Key_b's components $k_{b_0}, ..., k_{b_{q-1}}$ from **Alice** and computes $Key_b = \sum_{k=0}^{q-1} k_{b_k} (mod\ p)$;
9. **Alice** computes $Key = {Key_b}^{s_a} = g^{s_a \cdot s_b} (mod\ p)$;
10. **Bob** computes $Key = {Key_a}^{s_b} = g^{s_a \cdot s_b} (mod\ p)$;
11. **Alice** generates a cryptographic challenge and sends it to **Bob**;
12. **Bob** receives the challenge message, resolves it and replies to **Alice**;

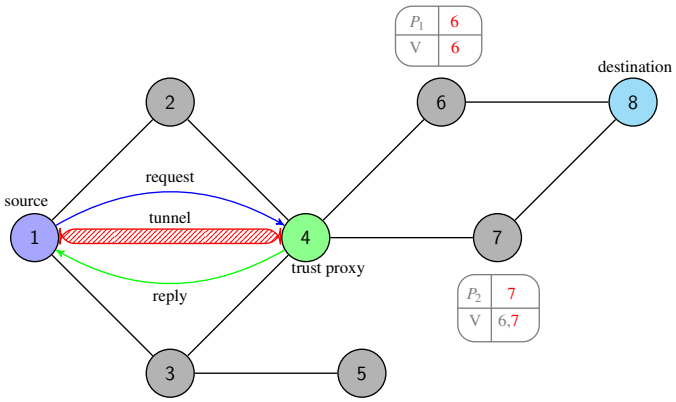


Figure 5. Key negotiation delegated to a trusted proxy (here represented by node 4).

Re-Authentication. Frequent key exchange causes both network overhead and latency. In order to overcome these issues, we propose to introduce re-authentication feature in our key exchange scheme. Indeed, in order to refresh tunnel and authenticate a correspondent node, a peer submit quite simply to its correspondent node a challenge to resolve. If the correspondent node successfully resolves the challenge and replies to the source node, then secure communication can be restarted over the old tunnel. Otherwise, key renegotiation will be needed.

Technically, challenge message is consisted of two random operands (O_1 and O_2) and one random operator *op* destined to be computed. Correspondent node then performs this arithmetical operation (O_1 *op* O_2) and sends the result, encapsulated within an encrypted packet, to source node.

Intrusion Detection Mechanism. *Trusted peers* can be used as proxies in order to join nodes that are unreachable via multipath (at least two disjoint paths). However, it could happen that a proxy node is compromised and can thus launch MITM attacks. In this scenario, the malicious node should be excluded from the list of *Trusted peers* and this event should be broadcast over the network.

Intrusion is detected when a peer fails to decipher data sent by its *legitimate* correspondent. In this case, *trusted peers* used as proxies for the negotiation phase will be repudiated from the list of *trusted peers* and this information will also be broadcast over the network.

4. Simulation Settings

To assess our approach, we use the *nem* simulator[4]. We use it for traveling along the paths between any pair of nodes in the studied network. In our experiments, different parameters are variable. These include the topology of the network, the size of the network, the proportion of network nodes that are considered to be compromised and the source and destination nodes. Each experiment follows a specific set of steps and is designed based on realistic network settings.

4.1. Experimental Steps

We succinctly present the steps that are carried out for the experiments:

1. *Definition of the network*: a P2P network is first created. In this step, we set the type of the topology (real map, synthesized topology such as Erdõs-Rényi, Internet-like, etc.) and the size of this network.

2. *Selection of the set of compromised nodes*: in the second step, we select a subset of X% nodes which will act as attackers. These nodes are supposed to coordinate their actions.

3. *Identification of source and destination nodes*: we then select, among the non-compromised nodes, a pair of source and destination nodes for the data exchange.

4. *Data packet transfer*: we launch the transmission by transferring a data packet through the shortest path towards the destination node. All intermediate nodes will be marked and may not be used for another packet between the same pair of source/destination nodes.

5. *Check for attacks*: at the end of the packet transfer, we check whether the packet was intercepted by an attacker.

6. *Use of alternative paths*: at this time, we start over from step 4, using the same source node but a different path to reach the same destination.

   (a) after we have selected a new path and started the routing process, if we arrive at a node where we can no longer move forward, we backtrack to the preceding node.

   (b) if we backtrack through all nodes until we reach the source node, we conclude that no other disjoint path exists in the network between the selected pair of source and destination nodes.

   (c) we also stop the search for alternate paths when we have tried all neighboring nodes of the source node. We also limit the search to a maximum of 10 disjoint paths between a given pair of nodes.

[4] http://www.labri.fr/perso/magoni/nem/

7. *Confirmation of the validity of the generated key*: we check whether the packet was potentially intercepted on all disjoint paths. If this is the case, then this attempt to generate a key is a failure. It is a success otherwise.

8. *Change of source/destination nodes*: we repeat the experiments starting from step 3 with a new pair of source and destination nodes.

9. *Change of compromised nodes set*: we repeat the experiments starting from step 2 with a new subset of compromised nodes. Basically, we change the percentage of attackers.

10. *Change of network settings*: we start over the experiments from step 1 with a new network topology and/or a new size value for the network.

4.2. Network Topologies

In our experiments, we have used two maps obtained by real Internet measurements. We have used an IPv4 map created in 2004 by traceroutes from multiple vantage points. It contains about 12.9k nodes and 60k links. We have also used a BGP-4 map created in 2010 by using the *route-views* BGP observer, which contains about 34k nodes and 70k links. We also use the Erdõs-Rényi [12] (ER) and the Magoni-Pansiot [16] (MP) models to build synthesized graphs of 4 different sizes (2.5k, 5k, 10k and 20k) for each model. The ER model creates random graphs while the MP model creates Internet-like graphs (i.e., following power laws and small world properties). In each map, we remove the tree part and keep the mesh only. We then proceed to execute 100 runs (i.e., each time with different attacker placements) for each scenario. Each run evaluates 1000 random pairs of source and destination nodes. Attackers are supposed to be coordinated (i.e., they know each other).

5. Simulation Results

We now describe the output of our experiments and the insights that they provide for assessing the performance of our approach. As suggested by our experimental steps, we explore the impact that each of the different parameters involved in the P2P network setup may have on the success of a key exchange which is defined as follows:

A key exchange is defined as successful if at least one subkey (i.e., key part) has not been intercepted by an attacker. This means that at least one disjoint path does not contain any attacking node and this guarantees that the key cannot be properly reconstructed by the attackers.

We then define the success rate as follows:

The success rate is equal to the number of successful key exchanges divided by the total number of key exchanges, the total consisting in the sum of the successful ones plus the unsuccessful (i.e., intercepted) ones.

5.1. Influence of Network Topology

The topology of a P2P network is a characteristic which influences various performance metrics. Because our approach of key distribution is at the overlay layer using coordinates from a hyperbolic plane, we are compatible with any topology for the underlying P2P network. However, the nature of this topology may affect the success of the key distribution scenario. Unless otherwise indicated, the settings of our experiments are: 10 disjoint paths and a maximum of 64 hops per packet.

Figure 6 depicts the success rates of key exchanges for various topologies, namely `Ring`, `MP` and `ER` topologies. We have plotted here the results of the experiments carried out with networks of 2.5k peers, and with a varying number of attackers defined as a % of the nodes.

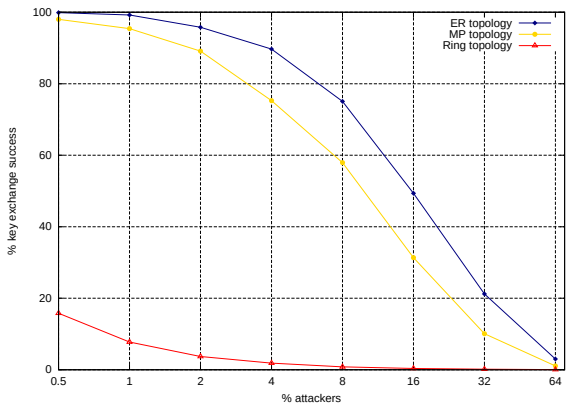
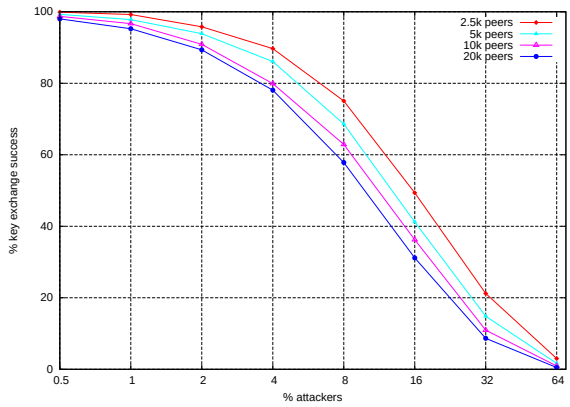


Figure 6. Success rate of the key exchanges for different network topologies.

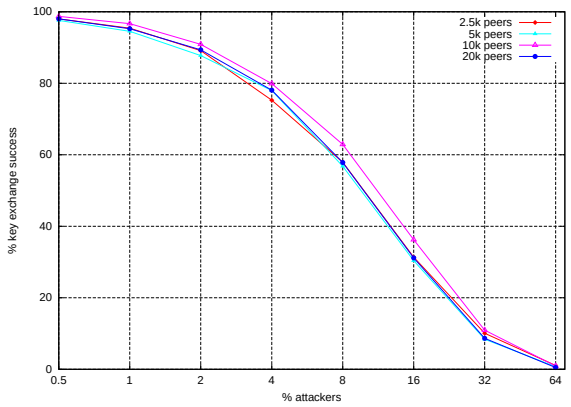
The graph shows that, for all topologies, the success rate steadily drops with the number of attackers. Beyond 60% for the number of attackers, it becomes virtually impossible to distribute keys in a safe way within the network. The `Ring` topology appears to perform poorly in a significant way. Indeed, while other topologies can still achieve close to a 100% key exchange success rate in front of 0.5% attackers, i.e., 100 infected nodes out of 20,000, the `Ring` topology achieves less than 5% exchange rate.

5.2. Influence of Network Size

We then proceed to investigate the performance in each network when varying the size of the network, on the one hand, and the degree to which the network is



a) ER topology



a) MP topology

Figure 7. Success rates for different network sizes.

infected, i.e., the proportion of attackers. Figure 7(a) illustrates the case of the er topology. The experiments were conducted with networks of alternative 2.5K, 5K, 10K and 20K peers. The ordering of the different curves suggest a strong correlation between the network size and the success rate of key exchanges, for all proportions of attackers.

In Figure 7(b) however, the graph shows that for the ER topology, the size of the network does not strongly influence the key exchange rate. Indeed, the evolution curves for the different network sizes are close and often intersect. Further, they are not ordered, with the top curve showing the evolution of a network of 10K peers, which is neither the smallest nor the biggest size considered.

Experimental results for the ring topology are depicted in Figure 8. As the network size grows, the success rate of key exchange drops. For large networks (e.g., 20K peers), even a small proportion of attackers (e.g., 2%) leads to virtually null performance in key exchange. We thus experimentally confirm with our implementation a well-known limitation of the Ring topology: multipath key exchange based on this topology is vulnerable to coordinated MITM attacks [23].

We can see that the network size has a small influence on the success rate of the key exchange and this depends on the network topology. In the ring topology this influence is the most important as increasing the number of nodes automatically increases the probability of having both paths intercepted.

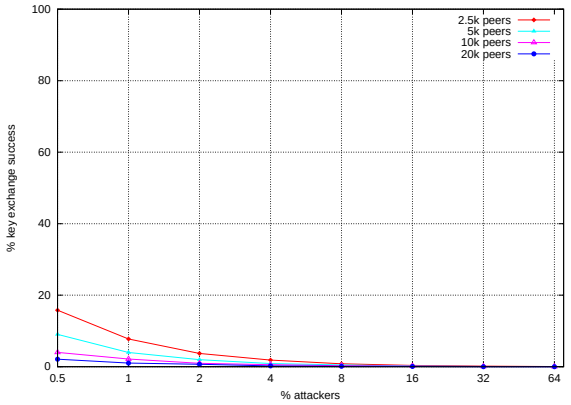


Figure 8. Success rates for different ring topology sizes.

5.3. Influence of the Chosen Number of Disjoint Paths

To reliably transfer a key between a source and a destination nodes, one must choose the number of disjoint paths to use for the different parts of the key (as the key split is done beforehand). We have performed experiments by varying the chosen number of disjoint paths from 1 to 10 for a network of 20k peers based on the ER model. The results are shown in Figure 9.

We observe that increasing the number of disjoint paths leads to an increase in the success rate of the key exchanges, which is expected, especially in the medium area where attackers are between 4% to 32%. We also notice a diminishing return when increasing the number of paths above 5 paths. There is a trade-off between increasing the success rate and decreasing the performances (as more paths means longer overall

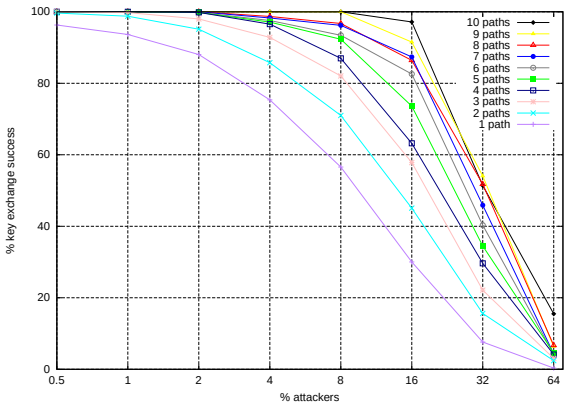


Figure 9. Success rates comparison by disjoint paths number (20k-node ER-model graphs with rerouting capped at 64 hops.

exchange time). Using from 2 to 5 paths already gives significant improvements on the success rate. The average number of available disjoint paths that can be found in a network between any given pair of nodes is thus important to exchange keys reliably.

5.4. Average Number of Available Disjoint Paths

Figure 10 shows the average number of disjoint paths found in 20k-node ER-model networks. We see that any source-destination pair has on average 7 paths. Many of them are intercepted on average but more than 16% of coordinated attackers are needed to be highly successful in intercepting all the subkeys, which makes the scheme efficient over this topology model.

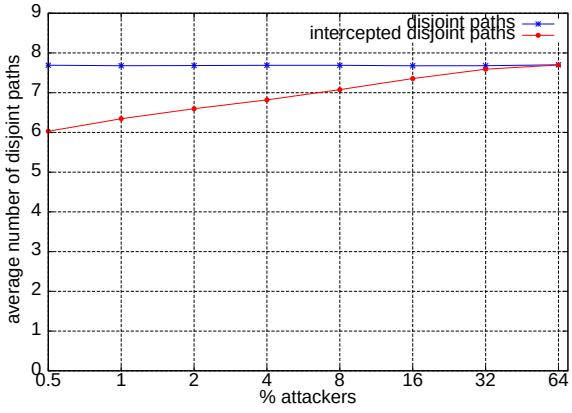


Figure 10. Average number of disjoint paths (total and intercepted) per source-destination pair in ER 20k-node networks.

Figure 11 shows the average number of disjoint paths in the 12.9k-node IPv4 map. We see that any source-destination pair has on average 3 paths, about half as what is found in the ER topologies. This is due to the fact that in the IP topology, the degree distribution of the nodes obeys a power law, which means that many nodes have only a few neighbors, while a few have a lot of neighbors. Despite this fact, much less paths are intercepted on average and more than 32% of coordinated attackers are needed to be highly successful in intercepting all the subkeys.

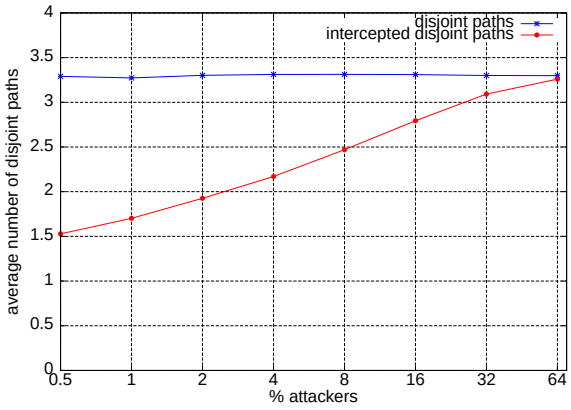


Figure 11. Average number of disjoint paths (total and intercepted) per source-destination pair in the IPv4 12.9k-node map.

The average number of available disjoint paths is thus highly dependent on the topology but also on the routing mechanism. Using a MP model topology with a strict greedy routing can lead to situations where those conditions render the key exchange inefficient or even impossible due to the lack of alternate paths.

5.5. Maximum Number of Hops

During a key exchange, the routing process requires that the identifiers of all nodes through which a packet transits must be added to this packet. Furthermore, each packet containing a subkey that leaves the source node, must contain the identifiers of all visited by previous packets carrying the first subkeys. Thus, it is important in practice to limit the number of hops between the source and the destination nodes, so as to prevent excessively huge packet sizes.

To measure the impact of the maximum number of hops, we have performed an experiment with two different values for the maximum. The experiments are based on a network of 20K peers and is built following an ER topology. Figure 12 illustrates the success rate of key exchange when the number of hops is capped at 64

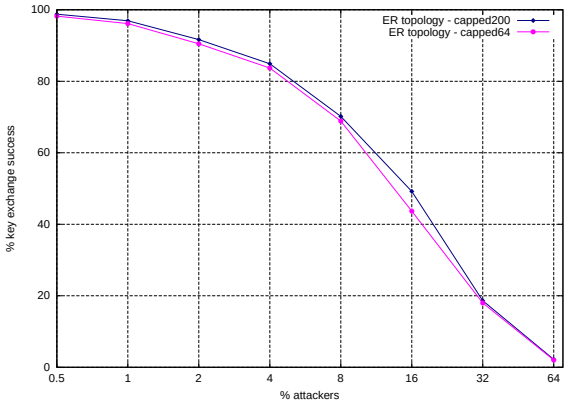


Figure 12. Influence of the maximum number of authorized hops.

and 200. We observe two curves that evolve similarly and are extremely close.

The number of hops has no significant influence on the success rate of the key exchange within the scope of the settings used in our experiments.

5.6. Influence of Routing Modes

We now discuss two modes of routing packets within the P2P network during a multipath key exchange process. In the *relax* mode, the process requires many disjoint paths that are relatively long. On the other hand, the *strict* mode corresponds to a reduced number of disjoint paths that are also short.

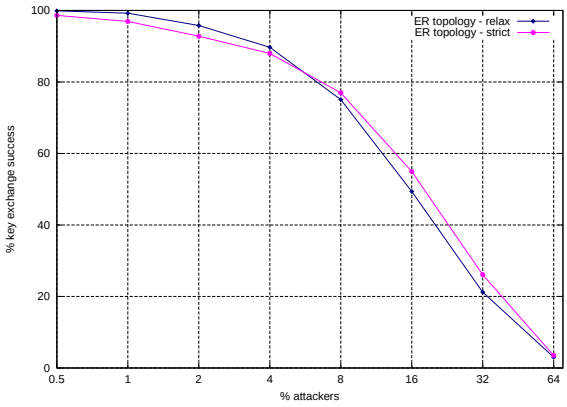


Figure 13. Comparison between the relax and strict routing modes.

Figure 13 depicts the results of our experiments in a network based on the ER topology. We observe that although the two curves are close, an interesting pattern emerges: for smaller proportions of attackers, the strict mode provides lower success rates for key exchange; for higher proportions of attackers, the relax mode is the one that leads to lower performances. This could be explained by the fact that when the number of attackers are small, a large number of paths (*relax mode*) increases the chance of avoiding them. On the other hand, when the number of attackers gets large, shorter paths (*strict mode*) are more effective.

Large numbers of disjoint paths and short paths favorably influence the success rates of the multipath key exchange scheme. The former works well in presence of small scale coordinated attacks while the latter is recommended for largely infected networks.

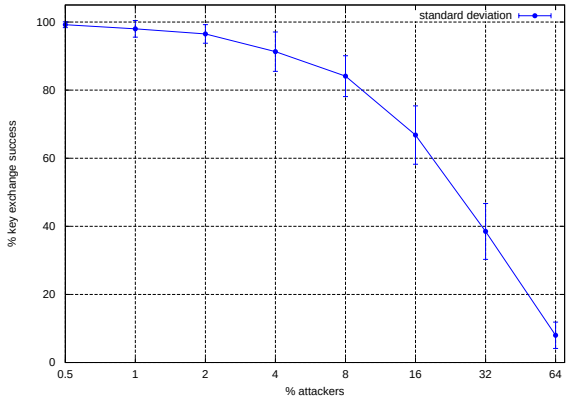


Figure 14. Standard deviation in the BGP-4 network map [17].

6. Security Analysis

In a multipath key exchange scheme, a malicious node that wishes to compromise a key being exchanged must be able to collect each of all key components routed over the network. Formally, when paths $\mathcal{P}_0, ..., \mathcal{P}_{k-1}$ are used to send several distinct subkeys from source $\mathcal{S}$ to destination $\mathcal{D}$, the only malicious nodes that could compromise the key should be located at the intersection of all paths. In other words, all the malicious node belong to set $M = \bigcap_{i=0}^{k} \mathcal{P}_i$ which represents the set of intersection points of all paths $\mathcal{P}_i$. $\mathcal{S}$ and $\mathcal{D}$ are obviously ignored in this set. Thus, when $\bigcap_{i=0}^{k} \mathcal{P}_i = \emptyset$ (*bigon criterion* is respected [11, Lemma 2.5]), then all paths are disjoint and any MITM attack attempt cannot succeed. In such a desirable case, there exists a k-connected subgraph between $\mathcal{S}$ and $\mathcal{D}$ in the network topology. When $|\bigcap_{i=0}^{k} \mathcal{P}_i| \geq 1$, there exists a real risk that MITM attacks will be committed on exchange transmitted between $\mathcal{S}$ and $\mathcal{D}$.

Consequently, the probability to have a MITM attack is estimated by $\sigma = \frac{|\bigcap_{i=0}^{k} \mathcal{P}_i|}{|\bigcup_{i=0}^{k} \mathcal{P}_i|}$ (where each path $\mathcal{P}_i$ is constituted of a set of consecutive hops from source $\mathcal{S}$ to destination $\mathcal{D}$). When all used paths are pairwise disjoint, the probability of *isolated* MITM attack (no *coordinated* MITM attack) is then: $\sigma = 0$ (i.e $|\bigcap_{i=0}^{k} \mathcal{P}_i| = 0$).

The number of distinct paths is also dependent on the source node degree. Thus, for a given *q-regular tree*, if q is a large number, then there is a probability to have several disjoint transmission channels. Nonetheless, despite the robustness of our multipath negotiation approach, cooperative (i.e., coordinated) MITM attacks, where several nodes maliciously cooperate to compromise a key, are possible. However, it is very hard, and excessively costly to execute such an attack in a real environment, especially in distributed systems where network topology changes dynamically.

In order to improve performances, re-authentication feature is introduced (see subsection 3.3). However, this challenge message used in this phase could be replayed. Furthermore, when a malicious node caches a challenge message, it can then create its copies and send them successively to target node. Thus, target node tries to resolves each challenge request because it does not know what packet is more fresh than other. Consequently, it will be rapidly saturated with requests from malicious node. Therefore, this causes a Denial-of-Service (DoS) attack.

In order to avoid such an attack from malicious nodes, a timestamp is assigned to each encrypted challenge message. Thus, the target node could distinguish between fresh packets and replayed packets.

Furthermore, during the key negotiation phase, all packets are exchanged in a clear text mode. Thus, traffic analysis attacks could reveal details about captured packets such as *sequence number* or *payload* which is nothing other than the transported subkey. Hence, multipath key exchange is needed to prevent the knowledge of all subkeys.

Figure 15 illustrates the slight decrease of key exchange success rate when the size of the network grows. In other words, the number of successful MITM attacks increases with the network size. As the number of attackers was set in our simulations to be proportional to the network size, this means that the size has indeed a small negative impact on the success rate. This is due to the consecutive average increase of the length of the paths themselves which increase their chance of being intercepted.

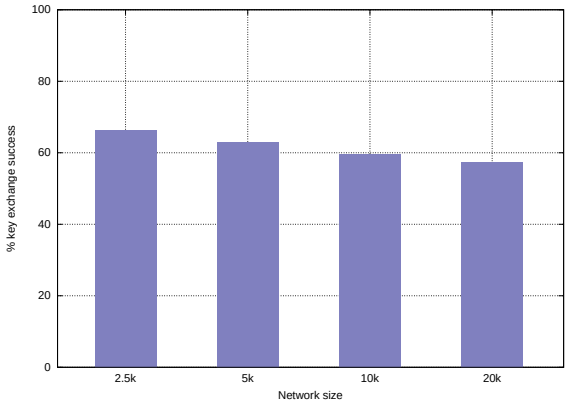


Figure 15. Variations of key exchange success rates for different network sizes (ER topology, capped by 64 hops max per path).

7. Related Work

Previous work have proposed several security infrastructures over fully decentralized or ad hoc networks [6, 7, 10, 15, 23, 27]. Although they are designed to be suitable in such environments, proposed approaches fulfill this gaol with more or less success and mostly with many caveats. In this section, we describe some models proposed in the literature to highlight the benefits of our approach.

Srivasta and Liu have relied on the Diffie-Hellman algorithm to deliver a solution that prevents threats in DHT networks [21]. Their scheme, however, remained sensitive to Man-in-the-middle attacks. Wang *et al.* have built a distributed PKI on top of the Chord structured overlay network [3]. They have used threshold cryptography to distribute the functionality of the PKI across the nodes of the DHT network. This Chord-PKI provides traditional PKI features such as certification, revocation, storage and retrieval.

Jiejun K. *et al.* propose to distribute certification authority functions through a threshold secret sharing mechanism [14]. In this system, private key is computed by k neighbor nodes and public key is derived from node identity.

Threshold cryptography is also used in identity-based key managements [7]. Nutshell, the key idea for identity-based cryptography is to define public keys derived from communicating nodes identities [20]. This method is really interesting however, the process of authentication could cause network overhead and the achieving of the value of threshold is not still guaranteed.

Takano *et al.* have designed a Multipath Key Exchange [23] similar to that proposed in our work. Their techniques however were designed to fit the Symphony and Chord P2P systems that are based upon

a ring topology. However, this inflexibility causes some drawbacks. Indeed, the proposed approach is based on clockwise/anticlockwise routing and this makes it sensitive to coordinated MITM attacks.

Jaydip Sen proposes a multipath certification protocol for MANETs that proceeds by broadcasting in order to discover the route between both source and destination nodes [19]. The key exchange protocol is based on this routing approach to retrieve the public keys of the nodes. However, broadcasting technique has proven that is not relevant in scalable networks such as fully decentralized P2P systems. Therefore,

Shehadeh E. *et al.* investigate secret key generation from wireless multipath channels [10]. The proposed protocol is based mainly on both the physical characteristics of the wireless channel and key predistribution schemes. This solution is implemented within physical layer.

8. Conclusion

P2P networks are self-organizing systems that do not need to resort to any central coordination point. The flexibility of such networks thus allows them to operate effectively on the Internet. Unfortunately, the inherent features that make them desirable also make them vulnerable to various security threats such as eavesdropping, modification and usurpation attacks.

In order to address the security challenges of P2P networks, we propose an improvement built upon our CLOAK architecture defined in our previous work [25] on one hand and a new approach for key exchange that generalizes a model proposed by Takano *et al.* [23] on the other hand. The benefit of using CLOAK instead of another DHT is that it does not use routing tables and does not impose any topology structure upon its peers. Our solution presented in this paper allows two peers from a P2P network to generate a common secret key in order to secure their communication without the need of a trusted third party when multipath is possible. This key generation is based on the Diffie-Hellman method with several subkeys being exchanged through several disjoint paths. Simulation results show that the probability of securely generating a key is above 90% when the percentage of coordinated attackers is lower than 2%. Depending on the topology type, the network size and the amount of attackers, the success rate can remain around 80% when the percentage of coordinated attackers is around 4%. As the size of P2P networks is typically several orders of magnitude (from thousands to millions of nodes), we expect our solution to be efficient because a few percent of coordinated attackers will result in hundreds of those attackers being needed in the network and that may not be feasible in practice.

For future work, we will consider adding peer authentication which for the moment only relies on the CLOAK DHT not allowing a peer to usurp another peer's name. We will also evaluate the time which is necessary to exchange a key. Although this happens only at the beginning of the communication and needs not be done after that even when interruptions happen, we will use event driven temporal simulations for investigating those delays.

References

[1] Daouda Ahmat, Tegawende Bissyande, and Damien Magoni. Towards securing communications in infrastructure-poor areas. In *5th EAI International Conference on e-Infrastructure and e-Services for Developing Countries*, 2013.

[2] Daouda Ahmat and Damien Magoni. Muses: Mobile user secured session. In *5th IFIP Wireless Days Conference*, pages 1–6, 2012.

[3] Agapios Avramidis, Panayiotis Kotzanikolaou, Christos Douligeris, and Mike Burmester. Chord-pki: A distributed trust infrastructure based on p2p networks. *Comput. Netw.*, 56(1):378–398, January 2012.

[4] Thomas Bohme, Frank Goring, and Jochen Harant. Menger's theorem. *Journal of Graph Theory*, 37(1):35–36, 2001.

[5] Cyril Cassagnes, Telesphore Tiendrebeogo, David Bromberg, and Damien Magoni. Overlay addressing and routing system based on hyperbolic geometry. In *Proceedings of the 16th IEEE Symposium on Computers and Communications*, pages 294–301, 2011.

[6] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):299–314, December 2002.

[7] Hongmei Deng, Anindo Mukherjee, and Dharma P. Agrawal. Threshold and identity-based key management and authentication for wireless ad hoc networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 - Volume 2*, ITCC '04, pages 107–, Washington, DC, USA, 2004. IEEE Computer Society.

[8] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[9] Antonio Gracia Dimitri Defigueiredo and Bill Kramer. Analysis of peer-to-peer network security using gnutella, 2002.

[10] Y. El Hajj Shehadeh, O. Alfandi, and D. Hogrefe. Towards robust key extraction from multipath wireless channels. *Communications and Networks, Journal of*, 14(4):385–395, 2012.

[11] D. B. A. Epstein. Curves on 2-manifolds and isotopies. In *Acta Math*, pages 15–16, 1966.

[12] Paul Erdős and Alfred Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5*, pages 17–61, 1960.

[13] Ken ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012.

[14] Jiejun Kong, Z. Petros, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Network Protocols, 2001. Ninth International Conference on*, pages 251–260, 2001.

[15] Hyeokchan Kwon, Sunkee Koh, J. Nah, and Jongsoo Jang. The secure routing mechanism for dht-based overlay network. In *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, volume 2, pages 1300–1303, 2008.

[16] Damien Magoni and Jean-Jacques Pansiot. Internet topology modeler based on map sampling. In *Proceedings of the 7th IEEE Symposium on Computers and Communications*, pages 1021–1027, 2002.

[17] Y. Rekhter. A border gateway protocol 4 (bgp-4). *RFC 4271*, http://www.ietf.org/rfc/rfc4271.txt ,january, 2006.

[18] J. Schafer and K. Malinka. Security in peer-to-peer networks: Empiric model of file diffusion in bittorrent. In *Internet Monitoring and Protection, 2009. ICIMP '09. Fourth International Conference on*, pages 39–44, 2009.

[19] J. Sen. A multi-path certification protocol for mobile ad hoc networks. In *Computers and Devices for Communication, 2009. CODEC 2009. 4th International Conference on*, pages 1–4, 2009.

[20] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[21] M. Srivatsa and Ling Liu. Vulnerabilities and security threats in structured overlay networks: a quantitative analysis. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 252–261, 2004.

[22] Jani Suomalainen, Anssi Pehrsson, and Jukka K. Nurminen. A security analysis of a p2p incentive mechanisms for mobile devices. In *3rd International Conference on Internet and Web Applications and Services*, pages 397–402, 2008.

[23] Y. Takano, N. Isozaki, and Y. Shinoda. Multipath key exchange on p2p networks. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pages 8 pp.–, 2006.

[24] T. Tiendrebeogo, D. Ahmat, and D. Magoni. Reliable and scalable distributed hash tables harnessing hyperbolic coordinates. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–6, 2012.

[25] Telesphore Tiendrebeogo, Daouda Ahmat, Damien Magoni, and Oumarou Sié. Virtual connections in p2p overlays with dht-based name to address resolution. *International Journal on Advances in Internet Technology*, 5(1):11–25, 2012.

[26] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of dht security techniques. *ACM Comput. Surv.*, 43(2):8:1–8:49, February 2011.

[27] Peng Wang, Ivan Osipkov, and Yongdae Kim. Myrmic: Secure and robust dht routing. Technical report, University of Minnesota, 2007.