Dynamic Routing in Covert Channel Overlays Based on Control Protocols

Peter Backs, Steffen Wendzel, Jörg Keller
Department of Mathematics and Computer Science, University of Hagen, Germany

***Abstract*—Covert channels aim to carry information in a way prohibited by the security policy and can be used to bypass censorship (e.g. by journalists). To establish secure covert channel communications, overlay networks with internal control protocols can be built.**

We present a design method for control protocols within covert channels. Our protocol design provides the advantage of space-efficiency (in comparison to existing control protocols) and the advantage of dynamic extensibility. We apply the protocol design to realize OLSR-based dynamic routing for covert channel overlays. Our algorithm provides different optimization means to maximize the covertness and the connection quality of the channel. The approach is validated by an extensible prototype.

I. Introduction

Covert channels (CCs) are hidden channels which are *not intended for information transfer at all* [1]. The intention of a CC is to hide the existence of an information flow that possibly violates a system's security policy [2]. CCs contribute to the free expression of opinions since they are useful to bypass censorship. Basically, network CC are divided into two classes: storage and timing channels [3]: While storage channels alter attributes of network packets (e.g. modifying unused bits), timing channels alter timings or the order of network packets to signal hidden information [4].

CCs have been a focus of research for decades. The topic was introduced in [1]; later it was described in [3] and [5]. In the following years, techniques were developed to deal with the problem of CCs, like the pump [6], covert flow trees [7], the shared resource matrix (SRM) methodology [8] and the extended SRM [9], timing channel elimination through program transformation [10], and machline learning-based timing channel detection [11].

Existing publications (e.g. [12], [13], [14]) describe how to implement CCs in network packet data and its timings. CCs also occur outside of TCP/IP networks, such as in business processes [2].

Research in the area of CC-internal control protocols (so called *micro protocols*, MPs) is required since such protocols can provide ways to enhance the known CC capabilities by, for instance, introducing dynamic protocol switches and adapting the channel configuration to changes in the underlay network [15]. These capabilities are of importance to provide CC users (e.g. journalists) advanced means to keep a communication undetected.

Small MPs for CCs already exist and can be found in popular tunneling tools like pingtunnel [16] as well as in [17], but these MPs have a static header design and are not as space-efficient as our MP, as we show in Sect. II-D.

A challenge in the context of covert communication is to keep data transfers as small as possible to decrease detectability [15]. To achieve such space-efficient communications, we present the design of a compact MP. This MP is based on the concept of *status updates*. A status update is a small data chunk, sent through a CC. Such a data chunk specifies a change in at least one setting (e.g. changing the destination address) of a CC. This concept helps to build more flexible and more space-efficient covert communications. A status update-based MP is used to control and dynamically adapt the covert communication between a sender and a receiver. It allows establishing of multi-hop CC routes. We are not aware of a previous MP which is both, dynamic (instead of a static header, it includes a dynamically configurable header for each new packet) and space-efficient (packet headers are designed to be as small as possible), as well as able to be used in conjunction with dynamic routing.

CC networks, as being overlay networks on the regular networks and as being similar to ad-hoc networks, comprise changing components and a changing infrastructure. While Szczypiorski et al. were the first authors to provide a dynamic steganographic overlay routing based on the random-walk algorithm in [18], we present a more-advanced approach based on the optimized link-state routing algorithm (OLSR). The dynamic routing algorithm is based on our previously mentioned concept of status updates. The presented routing algorithm is additionally optimized to generate as little overhead as possible to prevent raising attention.

Our routing algorithm takes into account the user requirements according to connection quality and covertness (called *Quality of Covertness*, QoC) and provides a maximized covertness, if these requirements can be met. We also propose to split the CC overlay network in so called *agents* and *drones*, i.e. passive and active routing components to optimize the covertness of the network.

We have developed an implementation of our status update-based MP to demonstrate its practicability and to validate the presented algorithm for dynamic CC overlay routing. Therefore, an extendible architecture called the *Smart Covert Channel Tool* was developed and successfully tested.

The remainder of this paper is organized as follows. Section II introduces the basic design principle of our MP. Section III presents the idea of an optimized CC overlay routing including the proof of concept implementation while

Section IV concludes.

II. Micro Protocols Based On Status Updates

A covert MP is a set of compact messages to be coded as covert data. A covert MP shares the few bits of space provided within the CC with the CC's payload [17]. Since a MP's data is placed within the hidden data of a CC, the MP depends on the security of the CC. The network CC's security highly depends on the number of modified bits within a network packet (the more bits are modified, the more anomaly can be caused). Thus, CCs often only provide a few bits of space for covert data within a single packet. Therefore, the tininess of a MP is mandatory requirement [17].

MP data can include a CC's meta information like covert payload size or payload offset, and can provide functionality like reliability over a set of multiple underlying protocols used simultaneously [17].

A. Status Updates

In standard network protocols (like IPv4) not all parts of a header are always used, which means that some parts of a header are transmitted even if they are unused at the moment (like sometimes the DSCP bits in IPv4). This problem is reduced if rarely used functionality is excluded in an optional header part. This is possible since decades in IPv4 via the "Options" part of the header. IPv6 includes a similar approach to build dynamic headers using the field "Next Header" that extends the IPv6 base header dynamically with additional functionality. However, IPv4/IPv6 headers contain unused information nevertheless, i.e. their design is not optimized for CCs.

A space-efficient concept for protocol headers is the compression for the *Serial Line Interface Protocol* (CSLIP) [19]. The concept of CSLIP (only transferring changing header parts for new packets) can be applied to network CCs. It is required to prevent changes in the network stack's behaviour for a network CC because their data must be placed in unobtrusive changes of network packets to remain undetected [17]. Thus, we cannot apply such CSLIP-like compression to the cover data but on the CC-internal MP. A comparison between our concept and CSLIP can be found in Sect. II-B.

Our status update design for a covert MP is similar to the IPv6 approach with the difference that the mandatory header part (called the "Type of Update" (*ToU*)) has only the size of a few bits which describe how to interpret the following bits (it is as if IPv6 would only consist of the "Next Header" field by default). The information that follows the ToU field depends on its value.

A simple status update-based covert MP could define the ToU values as in Tab. I while a full ToU list for a real implementation will be discussed later (cf. Tab. II). For example, if a sender would like to re-configure a CC peer between itself and a new destination peer D to forward all received packets to D, it would send the bits "01" followed by the destination address of D. Without status updates, each packet would have to contain a source address and a destination address, which results in bigger packets. By using status updates, these ToUs do only consume space if they are needed. It is important to understand that it is up to the implementer to choose a set of ToU commands as well as to define the behaviour of covert peers, since ToU commands depend on the required functionality of a CC (e.g. in case there is no "CONNECT" command, no ToU must be foreseen for the command). The presented ToUs and the presented CC peer behaviour are only examples. Fig. 1 illustrates a sample covert communication using status updates.

ToU	Meaning
00	SET SOURCE ADDRESS
01	SET DESTINATION ADDRESS
10	END OF UPDATES
11	PAYLOAD FOLLOWS DIRECTLY (followed by a length value if payload size varies)

TABLE I
A sample set of status update messages

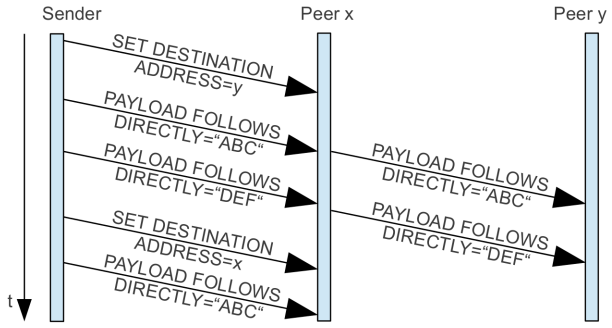

Fig. 1. A sample status update-based CC communication: A sender configures peer "x" to forward payload to "y". Afterwards "x" is configured to accept payload directly.

If a new connection is created, it must be configured. Therefore, a sequence of status updates (e.g. specifying the source and destination address) is needed which can also be obtained by a single static header. This means that there is *no* advantage of status updates for an *initial* covert connection setup (compared to existing MPs). On the other hand, status updates *are* an advantage compared to usual static headers (like IPv4) if few changes happen within an already established CC connection and within payload transfers since the header size of all non-initial packets is significantly decreased (because there is no need for an usual and comparatively large static header).

It is important to minimize the size of the MP header if its size is not negligible compared to the size of the covert payload. This can be done by optimizing the number of ToUs and compress them (e.g. using Huffman coding), see Section II-D.

B. Building Sequences

To reduce the total number of packets needed for a transmission, it is required to fill out all space provided within each

CC packet. For example, if a network packet would provide 24 bits of space and the ToU field would take only 2 bits while an overlay address value would require 6 bits, a sequence of ToU fields could look like in Fig. 2. The figure shows a network packet's covert data separated in 3 different areas. First, this packet includes two status updates: A new source address and a new destination address for all following packets. To signal that nothing follows, the set of ToU commands can include a value that specifies that no additional content follows (in Tab. I this is called "END OF UPDATES"). This is represented by the last two parts of Fig. 2. If no "END OF UPDATES" ToU is foreseen, the order of ToU values should be fixed as exemplified in Section II-D.

Fig. 2. A simple sequence of MP headers

Ray and Mishra introduced a MP that is split in two parts: the control part and the payload part [17]. This is different from our design. The ability of our MP to build sequences makes it more flexible and able to contain as many parts as needed. Of course, the set of parts contained in each packet can be different from earlier packets.

In comparison to the already mentioned CSLIP, our MP is more dynamic since it allows the same header component to occur multiple times per packet. CSLIP defines the included header components using a bit mask. Our status update sequences, on the other hand, define the type of update (ToU) right before each header part. This allows us to combine multiple packets in one packet using status updates. For instance, one can send the following sequence to a covert peer within only one packet: "SET DESTINATION", new address, "PAYLOAD FOLLOWS DIRECTLY", length=1, payload="A", "SET DESTINATION", new address, "PAYLOAD FOLLOWS DIRECTLY", length=1, payload="B". CSLIP is not designed to transfer as few packets as possible, but to reduce the size of the transferred packets. Our status update-based sequence allows us to place as much data within a single packet as possible, i.e. to transfer as few packets as possible with the goal to reduce the raised attention of the CC as described in [15].

C. Dynamic Underlying Protocol Change

A dynamic switch of the underlying protocol (e.g. switching from HTTP to DNS) within a CC can be based on status updates too if a ToU value is defined to initiate such a protocol switch. This is feasible since the presented MP does not depend on specific properties of underlying network protocols. Switching an underlying network protocol (e.g. transferring hidden data over HTTP instead of ICMP) can be initiated at any time. The determination of underlying protocols available for covert communication was presented in [20] and can be adapted to our approach. Using MPs, CC systems can additionally share underlying protocol usage information with each other and make them version-dependent, i.e. new CC MP versions can support additional network protocols to use for the covert communication [15]. This functionality enables users to upgrade existing CC infrastructure [15] and to bypass changed filter rules (e.g. a currently utilized underlying protocol is getting blocked and the CC therefore switches to another underlying protocol [20]).

D. Re-designing an existing MP

We re-designed the header of the MP presented in [17] as a status update-based MP. The given MP contains a static 8 bit header including a sequence number (2 bits), a data flag that indicates that payload is attached, an acknowledgement flag, an expected sequence number (2 bits) as well as two other flags (a start flag and a stop flag) to indicate when covert communication starts and ends (Fig. 3-a).

With status updates it was possible to reduce the average size of the existing MP header. This is done as follows: First, the protocol architect needs to find out which parts of the given MP header are not always required. In our case the last two bits (the start flag and the stop flag) are only required if the covert communication starts or ends. These two bits are candidates for removal from the default header.

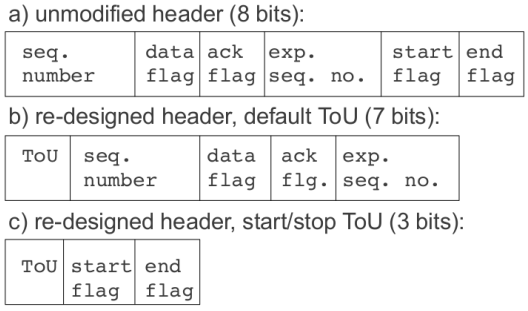

Fig. 3. a) The header from [17], b/c) Re-designed status update headers.

The second step is to build a list of ToUs. In our case there are two ToUs (Fig. 3-b/c): the default header (without start/stop flags) and a header that only contains the start/stop flag. To verify the usefulness of these new status updates, the protocol architect has to make sure that the size of the ToU field (in our case it has a size of 1 bit since 1 bit is enough to identify the two ToUs) is less than the number of bits removed from the default header. This is important because the ToU value must be placed in front of any status update. For the given MP, we were able to shrink the default header from 8 bits to 6 bits. By adding the additional ToU bit, the new default header has still a size of 7 bits instead of 8 bits. The second ToU (containing the start flag and the stop flag) requires 3 bits (one bit for the ToU and two bits for the two flags).

The third step is to verify the usefulness of a new status update design. This step fully depends on the MP used and aims to determine whether the new status update-based MP uses fewer bits in practice compared to the original MP. For

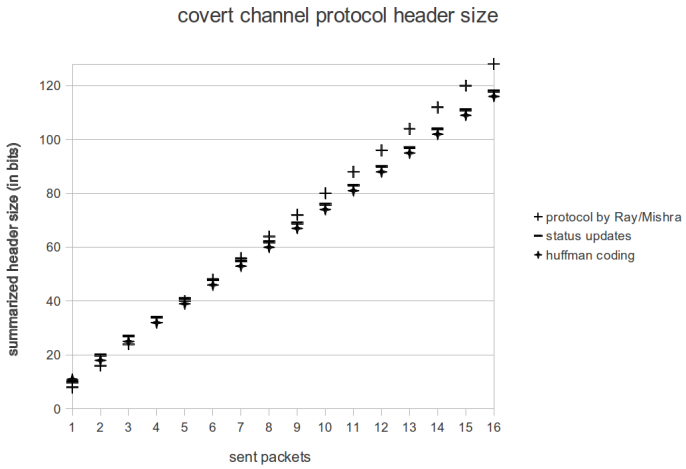

Fig. 4. Comparison of the summarized header sizes of the original MP and the re-designed MP. The new design is advantageous if a transmission comprises $\geq$7 packets (two ToUs, no Huffman) or $\geq$5 packets (three ToUs, Huffman).

example, for the given MP, the first as well as the the last packet of a transmission require a start flag respectively a stop flag. Therefore we need to send the default ToU plus the start-stop ToU in both the first and the last packet of a transmission. Both packets include the combined header which requires 7 bits + 3 bits = 10 bits if we use a pre-defined ToU order (see Section II-B where status update sequences are described). If we send two packets, the new MP requires 20 bits instead of 16 bits used by the unmodified MP.

Each packet of a transmission is shortened by 1 bit from 8 to 7 bits, and two packets of 3 bits each are added. Hence, if there are $n > 6$ packets, the amount of data is reduced by $n - 6$ bits. For large packet count n, the ratio of the total packet sizes approaches $7/8$ when comparing both designs.

Fig. 4 illustrates the summarized size of all headers for transmissions for different numbers of packets. If 7 or more packets are sent, the status update-based MP provides a better space-efficiency. If we split the start and end flag ToU in two ToUs and use a Huffman coding to gain a 1 bit ToU for the default header, we can improve the efficiency again, i.e., only 5 packets per transaction are required for a better space-efficiency.

However, the discussed MP of Ray and Mishra is designed for a *direct* communication between two CC peers and thus, does not require information about a connection's source and destination address. A MP used for a CC overlay in the context of dynamic routing must include source and destination information. Such address information increases a MP's size while being not always needed in each packet. Therefore, status updates will provide a better space-efficiency for MPs in dynamic routing overlays as in the discussed case of a direct communication.

III. Dynamic Routing in CC-Overlay Networks

CCs which are set up between two peers in a static fashion are not sufficient for complex covert communication scenarios. Drawing static CCs is far from the possibilities that can be realized in large and dynamic networks like the Internet.

A. Motivation of Dynamic Routing

Firstly, it seems obvious that forwarding a concealed message across multiple systems and multiple underlying network protocols of different kind can offer a much higher degree of covertness in comparison to a direct covert connection. By redirecting a message via multiple hops, the chance for an attacker to expose a complete hidden network communication is decreased. Multi-hop CC can even introduce completely new security goals: If the original sender and the destination of a covert message are veiled by communicating through several covert peers, the actual sender and receiver of the message are not revealed to an eavesdropper.

Furthermore, participants can join the network, go offline unexpectedly, crash or change their communication preferences resulting in overlay networks that may change their topology. Another reason for a change of topology may be to explicitly prohibit communication between two peers for reasons of covertness, i.e. to hide a relation between these two peers.

To enable multi-hop routing for dynamic covert overlay networks, a concept of route maintenance has to be introduced. In addition, the network routes should be chosen with reference to specific criteria for covertness and Quality of Service (QoS). Therefore, dedicated mechanisms for route plotting have to be introduced as well. Mechanisms for route maintenance and plotting are widely used in Internet technologies and play a central role. But there has been no attempt so far to effectively adapt these technologies for CCs.

In [21] a method is described that targets the dynamic routing in CC networks: TrustMAS (Trusted Multi-Agent Systems). The procedure describes a method for peer discovery and route plotting and a protocol for updating the routing information. Essentially, the process is based on random walk. For route plotting a sender of a covert message will decide by coin-flip if the message is either sent to its destination or to another peer that forwards the message. Then, the forwarding peer in turn decides by coin-flip if the message is to be forwarded to another peer or delivered to its destination. This procedure is repeated until the message reaches its destination. The concept also describes a process which determinates the neighbours of a peer in the overlay based on a random walk. Therefore the overlay topology is also random.

TrustMAS provides an easy way to manage dynamic CC networks. In addition, the algorithms scale well with large numbers of peers. However, routes plotted according to this concept are not optimal in terms of covertness and QoS. Besides, there is no possibility to ensure a certain degree of covertness or QoS. Furthermore the security objective of concealing the communication between certain peers (and communication via intermediate peers instead) cannot be met.

This is because the random walk algorithms and the random overlay topology are not capable of explicitly prohibiting direct communication between peers. Also, the covert messages are sent in an uncontrolled fashion from peer to peer resulting in high delay and jitter. Finally, since packets may also overhaul, measures must be taken to rearrange packets at the receiving site to their order of sending.

B. Quality of Service and Quality of Covertness

In many cases, the communication via CCs is very limited – an example would be a CC that wraps only a few bits of a covert message into the packets of an underlying communication which are sent only once every couple of hours, rendering small bandwidth and high delay or jitter.

For this reason, the quality of the connection must be taken into account at the time of route plotting. For example, there are different demands on the connection for the transmission of a video stream as for the transmission of one single password [15]. Thus, route plotting in CC networks must respect a certain set of parameters regarding end-to-end link quality that are commonly known as Quality of Service (QoS).

Another important aspect of CC communication is that CCs are based on the concept of security through obscurity, so no strict separation between "safe" and "unsafe" communication can be made. Instead, the degree of safety of the CC network is proportional to its covertness. CC technologies which cause a slight anomaly are less likely to be detected by an unauthorized party. On the other hand, CC technologies that cause a high anomaly increase the likelihood that an attacker discovers the CC network. Since there is a continuous spectrum of security, it must be left to the user which level of covertness in the CC network will be respected. The requirements for QoS are generally conflicting to those on the safety. To separate the requirements for QoS and security in the context of covert communication, the term *Quality of Covertness* (QoC) is introduced, summarizing the requirements for the covertness. QoC does not consider the overhead caused by the routing protocol. This overhead is based on the design of the routing protocol itself and the fluctuations of peers in the network. It cannot be regulated, unless the functionality of the CC network would be limited. QoC therefore concludes the covertness of a MP connection in terms of the security of the CC technologies used and the number of CC hops on the route.

C. Extending the MP with CC routing capabilities

If covert routes should be plotted respecting QoS and QoC, the peers must be aware of the other peers' CC capabilities and the CC network topology. CC capabilities of a peer describe which CC technologies a peer can handle. The CC network topology is the network graph that describes which peers may communicate directly with each other. Such routing information is propagated from one peer to another. Based on the demanded QoC, QoS, network topology and peer capabilities, a CC route can be plotted and established using status updates of the MP. We also use status updates for propagation of routing information. Thus, the list of ToUs of our MP has to be extended by message types that do not update the sending or receiving state of a peer, but the state of the CC network topology and the peer's CC capabilities. Besides the ToU used for establishing routes and sending messages, four additional ToUs for propagating routing information are required (Tab. II shows a complete ToU list for our implementation).

Type of Update	Meaning
REQUEST_PT_TT	Used by a peer to request the full peer table and topology table while bootstrapping.
RESPONSE_PT_TT	Response to REQUEST_PT_TT.
TT_LIST	A sequence of edges of the topology graph. Send on topology changes. Propagated according to MPRsel.
PT_ENTRY	A new or updated entry to the peer table. Send when a peer crashes, goes off, or joins the network, or changes CC capabilities. Propagated according to MPRsel.

TABLE II
TYPE OF UPDATE (TOU) VALUES FOR PROPAGATING ROUTING INFORMATION

D. Agents and Drones

If a CC network is only known by trusted users, it is unlikely that a participant will compromise it. So the approach is that a CC network consists of trusted users only. This requirement is linked to a considerable disadvantage: The CC networks are smaller because the number of users trusting each other is naturally small. But small CC networks pose the risk that not enough peers are available to the network to provide enough concealment. Yet worse, a CC network of only two participants would not be able to hide that these two systems actually communicate with each other. The number of possible routes in the CC network is also small, so that bottlenecks may occur, causing a severe anomaly in the underlying network.

In order to counteract the problems which result from small CC networks, it seems straight forward to increase the number of peers in the network. That could be realized if users run multiple peers in a CC network. However, this is difficult as the operation of several systems can often result in a considerable administrative and financial burden.

A better way to *artificially* increase the peer count is to use external systems as CC peers, which provide public available Internet services. Examples for such external CC peers are the servers of Google translator and the DNS servers:

Google translator servers may be used as a hop on a route in CC networks as described in [22]. A server of Google's translation service is sent a request to translate a specified website. The Google server loads the web page to be translated, translates it and sends the translation to the sender of the request. However, the original URL requested for translation may has a parameter string according to CGI (Common Gateway Interface) appended. This parameter string is handed over to the web server with the website to be translated. Now if the website is run by a peer in a CC network, then the CGI parameters may contain a CC message. This way a covert message can be transferred with Google translator

server as an external CC hop. This approach may work well with other services on the web, too.

Another way of including external systems in the overlay network is a special use of DNS. On request for resolving a DNS RR (*Resource Record*) that has expired, the DNS server has to update this RR at another DNS server. This RR update communication may also be used as a CC with the DNS server that receives the originating request as an external hop. An advanced implementation of CCs in DNS amongst others is described in [23].

These servers are not under the control of a CC user, and in particular, they do not execute any dedicated CC software. They can therefore not be regarded as peers with full functionality as those which implement a MP, but as passive participants. So they are referred to as *drones*. In contrast, peers running the CC software are called *agents*. Drones are limited in their functionality as a CC peer. They typically support only a few or even only one CC technology – namely the ones that can be embedded into the traffic of the web service provided. The difference between drones and agents is visualized in Fig. 5. Notwithstanding these limitations, drones are a comfortable way to expand a CC network with external systems as peers in a way that CCs can be established over a large number of CC-hops even on overlay networks with just a few users. Since drones can be external systems (i.e. public web services), they do not necessarily require to be aware of the covert communication.

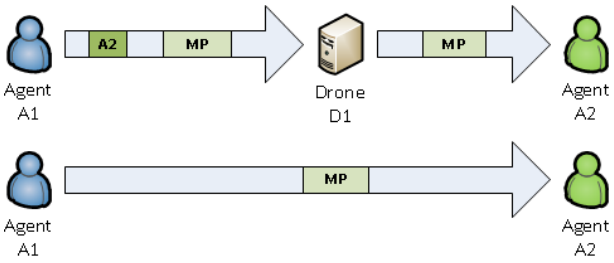

Fig. 5. Procedure of CC communication with and without a drone.

It is also conceivable to connect more than one drone in a row to form a CC. Since the sender has to encapsulate the covert messages into the underlay communication of each drone in the row, such connections will be considered as only one CC hop.

E. Routing

Routing in CC networks differs substantially from other routing techniques, such as on the Internet. The most important aspects amongst others are:

1) The routing overhead should be reduced as much as possible. In context of a CC this is necessary to keep a low profile.
2) The CC network topology may change faster than in less dynamic networks like the Internet. Links between peers can be dissolved or set by intention. Therefore a CC routing protocol must be capable of adapting quickly to topology changes. The raised attention by propagating topology change information must also be kept as low as possible.

These special requirements for dynamic routing in CC networks seem similar to those set for mobile ad hoc networks. It is therefore obvious that routing methods have been developed that could serve as a basis for a routing protocol in CC networks.

In addition to the above points, it is considered that the MP is used. That means the source node plots the route to the destination node and sets each peer along the path to status “Forward” or “Receive”. By these properties, the MP is to be classified as a source-routing protocol. Another aspect that implies a source-routing protocol is that the fulfillment of QoS and QoC requirements is controlled at a central point, and thus ensured more easily and without any additional communication overhead.

Since source-routing protocols are link-state routing protocols, mechanisms must be implemented to render the complete graph of the CC network in each peer – more precisely in all agents, since drones do not execute the CC software. The routing protocol should introduce as little routing overhead as possible. Optimized Link State Routing (OLSR) meets the above requirements. Furthermore, in a comparison of link-state protocols, OLSR is superior in terms of small overhead [24].

F. Smart CC networks

The presented method for routing in dynamic CC networks is called *Smart CC* (SCC). SCC is based on OLSR and additionally takes the specifics of CC into account, i.e. to support QoS and QoC and to respect the distinction between agents (full CC capabilities) and drones (reduced CC capabilities). Drones do not plot routes and therefore do not need any topology information, i.e. they do not need to participate in the link-state process of OLSR.

The fact that the CC network has drones which do not actively propagate link-state information could render the network graph in a state at which OLSR cannot operate properly. This happens when the graph of the CC network is discontinuous without drones. If that is so, the subgraphs are called agent clusters. Agent clusters are a special case when the propagation of routing information cannot be sent directly from one agent to another, but via a drone.

Nevertheless, the normal case is that routing information is resent to all neighbouring agents at the time of reception. Since the simple concept of flooding results in a great anomaly in the underlay network, routing information is propagated in a method similar to that of OLSR. OLSR introduces *Multi-Point-Relay selection* (MPRsel), a set of peers that is determined independently for each agent. Upon reception of routing information message, the message is resent if and only if the sender of the message is in MPRsel of the receiving peer. MPRsel is determined in such a way, that the number of resends is optimal in most cases.

In comparison to the propagation of routing information in OLSR therefore, two extensions are necessary:

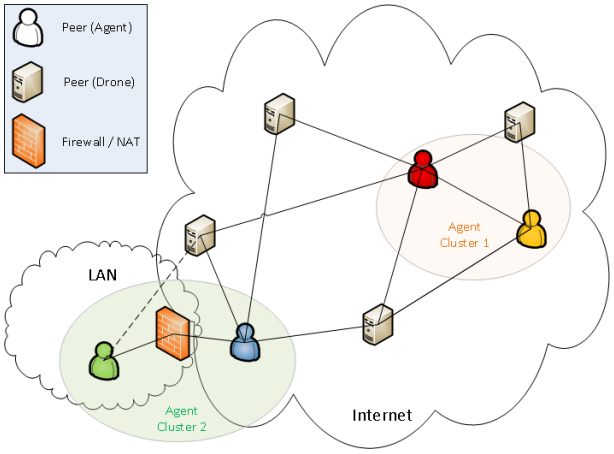

Fig. 6. A Smart CC network with agents, agent clusters and drones.

- The determination of MPRsel must mind that routing information is only sent to drones if it is necessary to connect two agent clusters.
- Routing information is broadcasted only to neighbouring agents, not drones.

The algorithm for plotting routes in CC networks based on the distributed link state information mainly consists of a function that is derived from Dijkstra's *Shortest Path First* algorithm in a way that it respects the users parameters in terms of QoS and QoC.

G. Proof of Concept Implementation

Based on the work in the previous sections, the *Smart Covert Channel Tool* (SCCT) was developed that enables dynamic routing in networks based on a MP.

The software was designed as a modular architecture. The modules either form a layer or an auxiliary module of a layer. Similar to the OSI reference model, these layers provide functionality abstracted from higher layers. This ensures that layers can be easily replaced without changing the entire program architecture. In addition, each component is a closed software module with a structured interface that makes the software easier to test and maintain in comparison to a monolithic architecture. According to the architecture, several test routines for unit testing of modules have been developed to help increasing the software quality. First of all these unit tests cover the complex calculation of MPRsel and broadcast neighbours. The plotting of covert routes using given QoS and QoC parameters is heavily tested, too.

Fig. 7 shows the modules of SCCT. The lowest layer is formed by the implementations of different CC technologies. Each of these technologies can be used to communicate via a CC of a certain kind. A CC technology is not capable of setting up multi-hop CC routes in the network, but builds a connection between two agents. The functionality of establishing multiple CC-hop routes is featured in the higher layers.

The central component of the application is the implementation of the MP as described. ToU for configuring peers

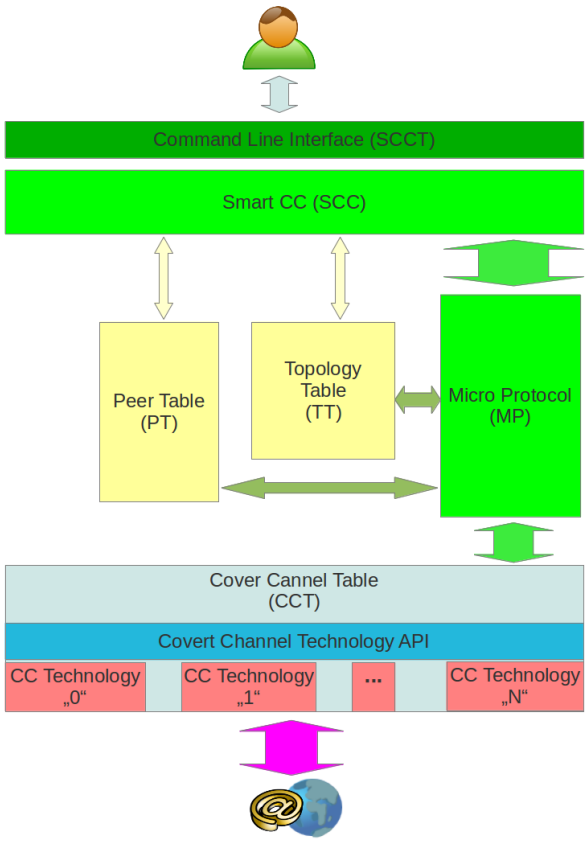

Fig. 7. SCCT software component model.

to forward an receive covert messages are provided. Route calculation and storing the network structure information are implemented in the auxiliary components topology table (TT) and peer table (PT). TT renders the adjacency matrix of the CC network graph and offers functionality to determinate MPRsel. The auxiliary component PT basically consists of a table describing the peers of the CC network and their properties.

The module Smart CC (SCC) provides an abstraction layer for the functionalities of the lower layers. At SCC layer it is possible to plot and establish a multi-hop connection via MP that meets a user-defined set of requirements for QoC and QoS with a single function call. In the layer SCCT a user interface is implemented that translates the user's commands into function calls to the other layers.

SCCT has been tested in serveral virtual network scenarios which simulate real CC networks. These scenarios include overlays with up to three agent clusters connected via two drones. The test networks had either been pre-configured in each peer to avoid initial routing overhead or built up automatically by learning the overlay topology from other peers.

H. Benefits of a CC API

The *CC Technology API* (CCT-API) provides a way for easy integration of different CC technologies into SCCT. As CCs and thus SCCT are based on security through obscurity,

introducing such an option is advantageous. CC technologies may be replaced, added or changed as needed for each CC network or user group and these own CC technologies remain secret within this group. Furthermore, users of the SCCT networks only need to implement new CC technologies according to the CCT-API and not the full functionality of a CC network program.

The CCT-API allows integration of CC technologies regardless of their classification. Thus, covert storage channels as well as covert timing channels may be used in SCCT. For indirect communication via drones, specific CC technologies can be developed and marked as such.

SCCT was developed as a proof of concept, which means the software's intention for production use is limited. However, the software meets the most important functions for dynamic routing networks in CC networks:

- Establishing and disconnecting CC-routes using the MP,
- support of drones,
- consideration of requirements for QoS and QoC,
- dynamically adding peers to the CC network,
- automatic propagation of routing information.

IV. Conclusion

We presented a design technique for covert channel-internal control protocols to optimize their space-efficiency as well as to enable a dynamic header design to reduce the raised attention of a covert channel. Our evaluation indicates that this space-efficient approach can perform better than an already space-efficient research protocol if at least five packets are sent per transaction. We expect larger space-efficiency improvements in comparison to non-optimized control protocols and control protocols for the dynamic routing in CC overlays since larger header components (such as addresses) can be removed from most packets if our *status update* approach is applied.

Based on this optimization technique, we realized the first OLSR-based dynamic routing for covert channel overlays. Our dynamic routing comprises the idea of *Quality of Covertness* (QoC) to optimize the covertness of a routing path as well as the usage of active and passive components (agents and drones) that either participate in the dynamic routing process or simply forward data without a necessary covert channel awareness.

We developed an extensible proof of concept implementation called the *smart covert channel tool* (SCCT) that is capable of utilizing different modular covert channel techniques simultaneously and of adapting to changes in the underlying network (e.g. administratively blocked protocols) using protocol switching. The supported techniques are linked to QoC and QoS values which are taken into account for the dynamic routing computation. The implementation has been verified by unit tests and simulated real-world scenarios consisting of several agents and drones.

In future work, we will compare the space-efficiency of our MP to non-research MPs as well (such as the *pingtunnel* protocol). We also plan to provide inter-operability for networks using different MPs via protocol translating gateway systems for CC overlays. Additionally, we plan the further development of SCCT to enable the handling of dead peers and the support of peers behind NAT.

References

[1] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[2] C. Wonnemann, R. Accorsi, and G. Müller, "On information flow forensics in business application scenarios," *IEEE COMPSAC Workshop on Security, Trust, and Privacy for Software Applications, IEEE*, pp. 324–328, 2009.

[3] Department of Defense, "Trusted computer system evaluation criteria (TCSEC)," Aug 1985.

[4] S. Zander, G. Armitage, and P. Branch, "Covert channels and countermeasures in computer network protocols," *IEEE Comm. Magazine*, vol. 45, no. 12, pp. 136–142, Dec 2007.

[5] J. P. R. Gallagher, Ed., *A Guide to Understanding Covert Channel Analysis of Trusted Systems (NCSC-TG-030)*. National Computer Security Center, Nov 1993.

[6] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The pump: A decade of covert fun," in *ACSAC*. IEEE Computer Society, 2005, pp. 352–360.

[7] P. A. Porras and R. A. Kemmerer, "Covert flow trees: A technique for identifying and analyzing covert storage channels," in *IEEE Symp. on Security and Privacy*, 1991, pp. 36–51.

[8] R. A. Kemmerer, "Shared resource matrix methodology: an approach to identifying storage and timing channels," *ACM Transactions on Computer Systems*, vol. 1, no. 3, pp. 256–277, 1983.

[9] J. McHugh, "An information flow tool for Gypsy - an extended abstract revisited," in *17th Annual Computer Security Applications Conference*, 2001, pp. 191–201.

[10] J. Agat, "Transforming out timing leaks," in *Proc. 27th ACM Symp. on Principles of Programming Languages*. ACM Press, 2000, pp. 40–53.

[11] S. Zander, "Performance of selected noisy covert channels and their countermeasures in ip networks," Ph.D. dissertation, Centre for Advanced Internet Architectures, Swinburne University of Technology, 2010.

[12] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday*, vol. 2, no. 5, p. http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/528/449, May 1997.

[13] T. G. Handel and M. T. Sandford, II, "Hiding data in the OSI network model," in *Proceedings of the First International Workshop on Information Hiding*. London, UK: Springer-Verlag, 1996, pp. 23–38.

[14] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: design and detection," in *ACM Conference on Computer and Communications Security*, V. Atluri, B. Pfitzmann, and P. D. McDaniel, Eds. ACM, 2004, pp. 178–187.

[15] S. Wendzel and J. Keller, "Low-attention forwarding for mobile network covert channels," in *Proc. of the 12th Conference on Communications and Multimedia Security*, ser. LNCS, vol. 7025. Springer Verlag, 2011, pp. 122–133.

[16] D. Stødle, "Ping tunnel – for those times when everything else is blocked," 2009, http://www.cs.uit.no/~daniels/PingTunnel/.

[17] B. Ray and S. Mishra, "A protocol for building secure and reliable covert channel," in *PST*, L. Korba, S. Marsh, and R. Safavi-Naini, Eds. IEEE, 2008, pp. 246–253.

[18] K. Szczypiorski, I. Margasinski, and W. Mazurczyk, "Steganographic routing in multi agent system environment," *CoRR*, vol. abs/0806.0576, p. http://arxiv.org/abs/0806.0576, 2008.

[19] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links (RFC 1144)," Feb 1990, http://www.rfc-editor.org/rfc/rfc1144.txt.

[20] F. V. Yarochkin, S.-Y. Dai *et al.*, "Towards adaptive covert communication system," in *PRDC*. IEEE Computer Society, 2008, pp. 153–159.

[21] K. Szczypiorski, I. Margasinski *et al.*, "TrustMAS: Trusted communication platform for multi-agent systems," in *Proceedings of the OTM 2008*, ser. LNCS, vol. 5332. Springer, 2008, pp. 1019–1035.

[22] M. Memelli, "g00gle crewbots," 2007, http://gray-world.net/projects/papers/gbots-1.0.txt.

[23] L. Nussbaum and O. Richard, "On robust covert channels inside DNS," in *24th IFIP International Security Conference*, ser. IFIP Advances in Information and Communication Technology, vol. 297, 2009, pp. 51–62.

[24] I. Glaropoulos, A. Makris, and B. Tighnavard, "Performance analysis of OLSR and comparison with OSPF and AODV," School of Electrical Engineering, KTH, Sweden, Tech. Rep., 2010.