This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Mutual Anonymous Communications: A New Covert Channel Based on Splitting Tree MAC

Zhenghong Wang[1], Jing Deng[2], and Ruby B. Lee[1]

[1]Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544, USA
{zhenghon,rblee}@princeton.edu

[2]Dept. of Computer Science
University of New Orleans
New Orleans, LA 70148, USA
jing@cs.uno.edu

***Abstract*—Known covert channel based on splitting algorithms in Medium Access Control (MAC) protocols requires the receiver's knowledge of the sender's identity. In this paper we present a new covert channel that does not have this restriction. In such a channel, multiple senders may operate independently without knowing each other, and the receiver can learn the transmitted information without knowing the identity of any covert sender *a priori*. These properties make the channel robust to malfunctioning senders, and more importantly help protect the secrecy of senders' identity which is essential for covert communications. We also analyze the capacity of our proposed covert channel.**

Keywords—security; covert channels; Medium Access Control.

I. INTRODUCTION

Covert channels, first introduced in [1], often refer to communication channels that are neither designed nor intended to transfer information. Covert channels usually exploit "legitimate" use of shared resources and operations of a system to leak sensitive information to someone who is not authorized to access it. For example, in a computer system with multiple security levels, a sending program (the *sender*) with a high security level can embed information into its usage of the system's CPU time and leak it out to a listening program (the *receiver*) with a low security level, bypassing all mandatory access control mechanisms. The sender can simply use as much CPU time as possible to send a bit '1' and use minimum CPU time to send a bit '0'. Other running programs in the system will experience longer delay if the sender sends a bit '1'. The receiver can therefore decode this information bit by measuring the delay it experiences.

Unlike traditional communication channels, a covert channel does not need to have a high capacity or transmission rate to be useful. In contrast, the stealthiness and resilience are much more important issues for covert channels. It should be hard for an auditor to discover if there is covert communications going on. More importantly, it is essential to ensure the secrecy of the sender's as well as the receiver's identity in all circumstances.

In past work, two types of covert communications were studied. One category exploits techniques that "hide" secret messages into existing "cover text", e.g., an image. Such techniques are usually referred to as *information hiding* and *steganographic* techniques. In covert communication over a network, it is a common technique to embed covert information into certain portions of network packets [3]. The other category of covert communications does not rely on any existing messages. Rather, some seemingly normal operations are exploited to interfere with the system so that the receiver can detect covert information from the system behavior. The CPU-time-based covert channel described above belongs to this category. A covert channel based on the splitting algorithms in Medium Access Control (MAC) protocol [2] is such a network-based covert channel. In this covert channel, the receiver needs to know the identity of the sender, even though the reverse is not needed.

In this work, we present a new covert channel that exploits splitting-tree algorithms in MAC protocols. Our new covert channel falls into the second category of covert communication channels, and has several salient features that significantly improve its stealthiness and resilience.

1) Multiple covert senders, without knowing the existence or identities of each other, can transmit covert information independently. The covert communication data rate and secrecy are improved (cf. sections III.D&E).
2) The receiver does not need to know the identity of the covert sender(s) *a priori*. We will show that such a difference has major effect on our covert channel's security performance (cf. section III.D).

The rest of the paper is organized as follows. In section II, the splitting algorithm in MAC protocol and the covert channel in [2] are reviewed. In section III, we first describe the basic idea of the new covert channel. We then discuss the detectability issue of the channel and present more advanced transmission techniques that improve transmission rate and communication secrecy. A brief capacity analysis is then presented. We conclude our work in section IV.

II. SPLITTING ALGORITHMS AND COVERT COMMUNICATIONS

A. Splitting-tree Algorithms

The network model is that of a time-slotted channel, with Poisson arrival, collision or perfect reception, *(0, 1, e)* immediate feedback. In our study, we assume that there are a fixed number of potential contending nodes. There are *m* covert senders and *n* normal contenders. We further assume that the data packets have a fixed length, L_{packet}.

Splitting-tree algorithms were designed to efficiently resolve collisions among active channel contenders. One example of such algorithms is the binary tree splitting algorithm. In this

This work is supported in part by DARPA and NSF Cybertrust CNS-0430487 and CNS-0636808.

algorithm, the colliding nodes are split into two subsets by randomly choosing either the left or the right subset to join. The nodes joining the left subset will send in the next time slot. The exact procedure depends on the outcome of the next time slot:

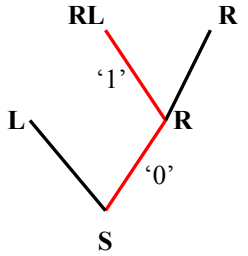
1) If the next time slot experiences collisions (*e*-state), the same procedure repeats for the new set of colliding nodes;
2) Otherwise, the next time slot experiences successful or idle state (1- or 0-state) and the right subset should transmit in the following time slot.

The period of time in which the collisions are resolved is called *Collision Resolution Period (CRP)*.

There are also improved schemes based on this basic splitting-tree algorithm. For example, one technique asks these nodes to split immediately if the next time slot is idle (0-state), since all colliding nodes have joined the right subset. In this work, however, we base our discussions on the standard binary splitting-tree algorithm.

B. Covert Channels based on Splitting Algorithms

As shown in [2], a node in the network can covertly send out a sequence of bits by choosing a specific path in the splitting tree in a CRP. Upon each collision, the sender intentionally, rather than randomly, chooses to join either the left set (if the covert bit to be sent is '1'), or the right set (if the covert bit is '0'). The receiver can be any node in the network or an external observer that knows the identity of the covert sender. At the end of each CRP, all nodes have transmitted successfully and the receiver sees the path that the sender has followed. The transmitted bits can then be decoded by looking at which subset the sender has joined at each *splitting point*, defined as the slot with collisions and further splitting is necessary. Figure 1 shows an example where the sender sends a bit string '01'.



	Transmitting set	Feedback
1	**S**	*e*
2	**L**	*1*
3	**R**	*e*
4	**RL**	*1*
5	**RR**	*1*

Figure 1. An example of covert transmission using splitting-tree algorithm

As shown in Figure 1, 3 nodes in the network contend for the medium in the CRP. The sender's path is highlighted with red color. The sender first joined the **R** subset to send a bit '0' and then joined subset **RL** (the left subset of set **R**) to send a bit '1'. Two covert bits were transmitted in this CRP. Note that the identity of the covert sender has to be known before the covert information is decoded. In our new covert channel, this is not needed.

C. Other Related Work

The notion of covert communication was first introduced in [1]. It was defined and analyzed, mostly in the context of computer systems, e.g., the Multi-Level Security (MLS) systems. A comprehensive survey of work in this area can be found in [4, 5]. To measure the significance of a covert channel, Shannon's theory of communications was often employed in the analysis of channel capacities [6-9]. Recent work also studied covert channels in communication networks that exploit the weakness of different communication layers [2, 3, 10]. Our work falls into this category.

III. A New Covert Channel

In this section, we present our new covert channel that exploits the splitting-tree algorithm. This channel allows multiple anonymous senders to simultaneously transmit covert information to the anonymous receiver, resulting in more robust communication and higher transmission rate. A salient feature of our new covert channel is that the receiver and the covert senders do not need to know each other's identities *a priori*. The senders do not even need to know the existence of their peer senders. This makes the channel survivable under extreme circumstances. For example, if a sender is captured and all information it knows is revealed, other senders and the receiver are still safe.

We assume that every sender keeps an identical copy of the covert text to be sent, and these senders are initially synchronized at a certain position in the text, e.g., the first bit of the whole sequence.

In the rest of the section, we first present a Basic Covert Transmission (BCT) technique that sends one bit in each CRP. Then an Advanced Covert Transmission (ACT) technique with higher transmission rate is proposed. We discuss the detectability issue of these techniques, and then we present an Undetectable Covert Transmission (UCT) technique.

A. Basic Covert Transmission (BCT) Technique

```
IF collision in the 1st slot THEN
  FOREACH sender t_i DO
    x = rand();
    b_i = getbit(t_i);
    IF bi == '1' THEN
      IF x < p_b THEN
        join_L(t_i);
      ELSE
        join_R(t_i);
    ELSE
      IF x < p_b THEN
        join_R(t_i);
      ELSE
        join_L(t_i);
  END
```

Figure 2. Transmission procedure of BCT

Figure 2 shows the transmission procedure of BCT. *getbit*(t_i) returns the bit to be sent at the current position in the covert text of sender t_i and moves the pointer to the next bit; *rand()* returns a random number x that uniformly distributes in [0,1); p_b is a pre-defined *biased probability*, $0 < p_b < 1$.

At the beginning of each CRP, if there is a collision in the first slot, the transmission set needs to split. Note that although in Figure 2 the operation for a sender is in a FOREACH loop, all operations are executed independently by each sender in parallel rather than in serial. Depending on the value of the covert bit, each sender joins either the L subset or the R subset with a biased probability p_b (e.g., 0.95). This would make the L

subset larger if a '1' is sent or R larger if a '0' is sent. The receiver can then decode the transmitted bit based on the sizes of the L and R subsets. Note that the random behavior of the normal nodes may corrupt the above relationship with certain probability and make the channel erroneous.

Hard decision can be made in the reception procedure, i.e., if subset L is larger than R, a '1' bit is decoded. Otherwise a '0' bit is decoded. Other more sophisticated techniques can be used here to achieve more reliable transmission, e.g., channel coding and/or soft decision can be applied here to overcome errors. However such detailed discussion is out of the scope of this paper.

B. Advanced Covert Transmission (ACT) Technique

The covert information transmission rate using BCT is very low: one bit transmitted in each CRP. We propose here an Advanced Covert Transmission (ACT) technique that can send multiple bits in each CRP. The basic idea is to ask covert senders to transmit one covert bit at each splitting point in a CRP, using the BCT technique repeatedly. In order to maintain synchronization among all covert senders and between the senders and the receiver, everyone needs to advance one bit in the covert text whenever there is a splitting point, regardless of whether it participates in the splitting operations.

```
FOREACH splitting set S_k DO
  FOREACH sender t_i DO
    b_i = getbit(t_i);
    IF t_i ∈ S_k THEN
      x = rand();
      IF bi == '1' THEN
        IF x_i < p_b THEN
          join_L(t_i,S_k);
        ELSE
          join_R(t_i,S_k);
      ELSE
        IF x_i < p_b THEN
          join_R(t_i,S_k);
        ELSE
          join_L(t_i,S_k);
  END
END
```

Figure 3. Transmission procedure of ACT

Figure 3 shows the transmission procedure of ACT. Upon a collision, the corresponding transmission set (e.g., S or L in Figure 1) needs to split. If a sender belongs to this set, it will try to send the bit in a similar way as in BCT. Other senders sit out during this splitting operation but they need to skip one bit to keep synchronization. After the CRP is completed, all senders "send" the same number of covert bits (some of the bits are actually skipped, though at different positions for different senders). The number of bits that have been sent equals to the number of splitting points in the splitting tree.

C. Detection of Senders and Stealthiness of the Channel

As the receiver is a pure passive listener during both the transmission and reception operations, it is undetectable even if the auditor is aware of the covert communication. However, with BCT or ACT it is still possible that a covert sender or even an external observer can figure out the identities of the covert senders. To do so, a covert sender simply looks for the nodes that go to the same branch as itself with a probability higher than 0.5. An external observer can detect the covert senders by looking for nodes that go more frequently to the same branch of the splitting tree. It is not difficult to show that the larger p_b, the easier it is to find the covert senders. Furthermore, with larger p_b, senders will be more likely to join the same subsets and therefore cause more collisions. This will artificially lengthen the CRP and raise suspicions. It is of course possible to improve the covert channel's stealthiness by employing a p_b closer to 0.5. However, this leads to channel capacity degradation. With ACT, a trade-off has to be made between the stealthiness of the covert channel and the capacity.

D. Undetectable Covert Transmission (UCT) Technique

To prevent an arbitrary observer other than the receiver from detecting the covert communications and/or the identities of the senders, a secret shared by the senders and the receiver is added into the scheme. One can detect the covert communications or may identify the covert senders only if he knows the secret.

However, simply letting all senders know the common secret will significantly reduce the resilience of the channel: if a sender is captured, it can be disassembled and the secret will be exposed, leading to easier discovery of all other senders. Below we present our improved scheme that overcomes this problem.

***Assumptions*:**

1) Each node in the network has a unique ID;
2) The receiver is always safe.

Given that most networks use some sort of unique ID for each node to address the network packets, the first assumption should not impose extra restriction on existing networks. We assume that the receiver is safe because it is a purely passive listener and hence undetectable. Furthermore, unlike the senders, who are network nodes and often accessible for testing and investigation by the network administrator, the receiver can be an external listener that is out of the investigator's control.

***Secret distribution*:** a secret K is shared among all covert senders and the receiver but is stored in different forms.

1) Receiver: the receiver keeps a copy of K in its original form since it is considered safe.
2) Sender: each sender t_i keeps $C_i = \text{HASH}(\text{ID}_i||K)$, where HASH() stands for a one-way hash function and || is the concatenation operator.

The one-way property of the hash function ensures that even if C_i is revealed, it is impossible to calculate K based on C_i and ID_i. Any good cryptographic hash function such as SHA and MD5 can be used to generate C_i. By doing this, even if a sender is captured and C_i is discovered, K remains secret.

***Operations*:**

1) Sender: the sender's operation is identical to ACT/BCT except that the covert bits are pre-coded before transmission. The pre-coding scheme is simple: XOR the original covert bits with a sequence of random bits generated based on a common random source and the node's secret C_i. Detailed steps are shown in Figure 4.
2) Receiver: the receiver first needs to decode the transmitted

bits into its original form for each node (assuming all nodes are senders). The final received bits are then derived based on the decoded bits for all nodes. Figure 5 shows the detailed operations.

STEP 1: At the beginning of a CRP, obtain a random number *Seed* based on a common random source.
Seed = common_rand();

STEP 2: Generate a sequence of random bits *Bitseq* based on *Seed* and its secret C_i, using a pseudo random number generator *PRNG()*.
Bitseq= PRNG(seed||C_i);

STEP 3: Pre-code the original covert bits text with *Bitseq*.
newtext = Bitseq XOR text;

STEP 4: *Send the bits using ACT or BCT*
ACT(newtext); (or BCT(newtext);)

Figure 4. Sender's Operations

STEP 1: Once a CRP is complete, decode the transmitted bits $TXbits_i$ for each node N_i based on the splitting tree *T*.

FOREACH *node* N_i **DO**
  $TXbits_i$ = *obtain_bits*(N_i, *T*);
END

STEP 2: For each node N_i, decode $TXbits_i$ into the original bit sequence *Bits[i];*

Seed = common_rand();
FOREACH *node* N_i **DO**
  C_i = *HASH* (ID_i||*K*);
  $Bitseq_i$ = *PRNG*(*seed*||C_i);
  Bits[i] = $TXbits_i$ *XOR* $Bitseq_i$;
END

STEP 3: Derive the final received bits *text* based on *Bits*
text = decode(Bits);

Figure 5. Receiver's Operations

The common random source should be visible to all senders and the receiver. For instance, it can be obtained from the network itself during the normal communication, e.g., the content of one or more data packets transmitted in the *previous* CRP, e.g., the first transmitted packet, the last transmitted one, or packet(s) that can be determined using any predefined (and possibly public known) rules. With this common random source and the secret C_i, each sender can generate a unique random bit sequence to pre-code the covert bits. The receiver can generate exactly the same sequence of random bits for each node based on the node ID and the secret *K*. It is possible that one can force all nodes to send non-random data packets to remove the randomness and detect the covert channel. However, since the senders are also nodes in the network, they can stop sending out bits to protect themselves once they receive such commands.

In step 1 of the receiver's operations, once the CRP is completed, the bits (the pre-coded bits) that a node has transmitted can be determined: at the splitting point where it participated in set splitting, a bit '1' or '0' can be determined based on whether it joined the L subset or the R subset. At other splitting points, a symbol '*e*' is put into the bit sequences, meaning that the bit has been skipped by this node.

In step 2, for each node the receiver needs to generate the same random bit sequence that was used in pre-coding the covert bits during the sender's transmission procedure (see Figure 4 -- STEP 2). To do so, it first computes C_i for each node using the secret *K* and the node's ID_i: $C_i = \text{HASH}(ID_i||K)$. It then computes the random bit sequence in the same way as the sender does in Figure 4. The original covert bits can then be recovered with XOR operations. Note that the bit sequence obtained in step 1 may contain '*e*'. Any XOR operation over '*e*' will still generate an output of '*e*'. The obtained bit sequence for each node therefore is a series of '1', '0' and '*e*'.

The receiver then computes the sizes of the L subset and the R subset for each splitting point. This can be done by the decode() function shown in Figure 6. For a given splitting point, the numbers of nodes that join the L subset and the R subset are accumulated in variables *size_L* and *size_R*, respectively. The final decoded bit can then be determined based on these two numbers. Figure 6 shows a decoder using hard decision.

```
FUNCTION decode (Bits)
  /* number of bits transmitted in the CRP */
  M = number of branching points in splitting tree;
  FOR j = 1 TO M
    /* initialize counters */
    size_L= size_R = 0;

    FOREACH node Ni DO
      b = Bits[i][j];
      IF b != 'e' THEN
        IF b == '1' THEN size_L++;
        IF b == '0' THEN size_R++;
    END

    IF size_L >= size_R THEN
      text[j] = '1';
    ELSE
      text[j] = '0';
  END
```

Figure 6. An example decode function

***Security Analysis*:**

Stealthiness of the channel: In UCT, each sender in the CRP generates a unique random bit sequence with which the covert bits (identical to all senders) are encoded into the actual transmitted bits. If the *PRNG()* function can generate independent sequences with different seeds, the covert senders will not join the same subset with high probability. This makes the sender nodes indistinguishable from normal nodes, which prevents an external observer from detecting the covert communications. In practice, cryptographically secure pseudo random number generators (CSPRNG), e.g., the AES-based or SHA-based PRNG, can be good candidates of the *PRNG()* function.

Detection of the covert senders: For a covert sender t_i, although it knows C_i, the property of the *HASH()* function ensures that it cannot derive the secret *K* or C_j of any other sender. This prevents the sender from correctly decoding the transmitted bits of any other sender, which makes it impossible to find out the peer senders. Similarly, if any one of the senders

is captured and its secret C_i is discovered, the secrecy of the identity of other senders will not be compromised.

Discussion:

Initial synchronization: In the above discussion we assume an initial synchronization of all covert senders. Although a system clock should of course help achieve this synchronization, we do not assume this since it may be unrealistic. Similar to the way we obtain the common random source from the normal operations of the network, the senders may synchronize themselves with some common events such as a certain byte of a successful packet on the channel or the hash of it being equal to a predefined value. If such an event occurs in the current CRP, all senders start to transmit in the next CRP. A sender may fail to synchronize with others, but this does not mean the covert communication will fail, as we explain below.

Malfunctioning nodes: A malfunctioning sender may lose synchronization and/or behave incorrectly during the covert transmission. However, since there is no cooperation among peer senders, this will not affect other senders. From the receiver's perspective, a malfunctioning sender simply behaves like a normal node, which reduces the number of effective covert senders by 1 and degrades the channel capacity slightly.

Discovery of the senders' identity by the receiver: The receiver does not need to know the identities of the covert senders *a priori*. It will treat all nodes in the network as covert senders, decode their messages, and regenerate the covert text using majority voting. In this process, the senders' identities will eventually be disclosed to the receiver. Since the receiver is undetectable, such disclosures do not change the stealthiness of our covert channel. Note that this is more flexible and secure than requiring the receiver to know the identities of the senders *a priori* [2]. For instance, any covert sender can join in or quit the covert transmission without the need to keep the receiver updated.

E. Capacity Analysis

The channel with BCT can be modeled as a discrete time memoryless channel (DMC). The input alphabet of the channel is {'0','1'}. The output alphabet is a set of ordered pairs (L,R) where L and R are the numbers of nodes that joined in the left subset and right subset, respectively. It is rather straightforward to compute the channel matrix, given the number of covert senders, the number of normal nodes, and the biased probability p_b used in Figure 2. The capacity of such a channel can then be easily calculated. The analysis of the channels with ACT and UCT is rather complicated. Due to the limited space, we omit the detailed derivation of the capacity bounds, which can be found in our full technical report [11].

Our results show that the capacity of the channel with BCT is low. For example, in a network with 30 nodes among which 15 are covert senders, when a $p_b = 0.9$ is used, the channel capacity is approximately 0.96 bit/CRP or 0.012 bit/slot. In contrast, using ACT the channel capacity is improved significantly. With the same network parameters, the upper bound of the capacity is approximately 36 bits/CRP or 0.4 bit/slot. In a channel with UCT, a more biased p_b can be used since the covert communication is undetectable. Our result shows that in the same network with p_b=1, the upper bound of the capacity is approximately 0.35 bit/slot, which is only slightly lower than the capacity of the channel with ACT.

IV. Conclusions and Future Work

In this paper, we have proposed and investigated a new covert channel based on the splitting tree algorithms in MAC protocols. The covert information is embedded into the covert senders' behavior in choosing which subset to join during a splitting operation. Unlike other known covert channels based on splitting tree algorithms, this channel allows multiple senders to participate in the covert communications, improving the robustness and transmission rate of the channel.

One salient feature of this covert channel is that it does not require the receiver's knowledge of identities of the covert senders. Furthermore, all senders operate without knowing the identities of other senders or that of the receiver. This mutual anonymity significantly improves the resilience of the channel: even if part of the channel is compromised, e.g., one or more covert senders are captured and their information is exposed, the rest of the channel remains safe. We have analyzed the transmission techniques involved in this channel and discussed the detectability issue. We further proposed a UCT scheme that enables undetectable covert communications. The capacity of the channel with different transmission techniques is also analyzed and calculated.

Future work will involve further investigation in other transmission schemes that allow more robust and faster covert communications, and countermeasures to mitigate these covert channels.

References

[1] B.W. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, vol. 16, issue 10, pp. 613-615, October 1973.

[2] S. Li and A. Ephremides, "A Covert Channel in MAC Protocols Based on Splitting Algorithms," in *Proc. Of IEEE Wireless Communications and Networking Conf. (WCNC)*, March 2005.

[3] T. Handel and M. Sandford, "Hiding Data in the OSI Network Model," in *Proc. of the First Int. Workshop on Information Hiding,* Cambridge, U.K., May-June 1996.

[4] National Computer Security Center, "A Guide to Understanding Covert Channel Analysis of Trusted Systems," *NCSC-TG-30*, November 1993, available at http://www.radium.ncsc.mil/tpep/library/rainbow/.

[5] John McHugh, "Covert Channel Analysis: A Chapter of the *Handbook for the Computer Security Certification of Trusted Systems*," December 1995, available at http://chacs.nrl.navy.mil/publications/handbook/.

[6] J.K. Millen, "Covert Channel Capacity," in *Proc. of the IEEE Symposium on Research in Security and Privacy*, pp. 60-66, April 1987.

[7] I.S. Moskowitz, S.J. Greenwald, and M.H. Kang, "An Analysis of the Timed-Z Channel," in *Proc. of IEEE Computer Symposium on Security and Privacy*, pp. 2-11, May 1996.

[8] J.K. Millen, "Finite-State Noiseless Covert Channels," in *Proc. of the Computer Security Foundations Workshop II*, pp. 81-86, June 1989.

[9] Zhenghong Wang, and Ruby B. Lee, "Capacity Estimation of Non-Synchronous Covert Channels," in *Proc. of the 2nd Int. Workshop on Security in Distributed Computing Systems*, pp. 170-176, June 2005.

[10] K. Szczypiorski, "HICCUPS: Hidden Communication System for Corrupted Networks," in *Proc. of the 10th Int. Multi-Conf. on Advanced Computer Systems*, pp. 31-40, October 2003.

[11] Zhenghong Wang, Jing Deng and Ruby B. Lee, "Mutual Anonymous Communications: A New Covert Channel Based on Splitting Tree MAC," *Princeton University Department of Electrical Engineering Technical Report CE-L2006-011*, July 2006.