SURREAL: SecURe coveRt peEr communicAtion over BitTorrent protocol

Avinash Srinivasan  
Computer and Info. Sciences  
Temple University  
Philadelphia, PA 19121  
Email: avinash@temple.edu  
Fellow – National Cybersecurity Institute

Jie Wu  
Computer and Info. Sciences  
Temple University,  
Philadelphia, PA 19121  
Email: jiewu@temple.edu

Richard Joy and Hatoon Aldarrhab  
Volgenau School of Engineering  
George Mason University,  
Fairfax, VA 22030  
Email: [rjoy, haldharr]@gmu.edu

Abstract—Today, peer-to-peer (p2p) networks have risen to the top echelon of information sharing on the Internet. It is a daunting task to prevent sharing of both legitimate and illegitimate information such as – music, movies, software, and child pornography – on p2p overt channels. Considering that, preventing covert channel information sharing is inconceivable given even its detection is near impossible. In this paper, we describe SURREAL – a technique for covert communication over the very popular p2p BitTorrent protocol. Standard BitTorrent protocol uses a 3-step handshake process and as such does not provide peer authentication service.

In SURREAL, we have extended the standard handshake to a 6-step authenticated covert handshake to provide peer authentication service and robust peer anonymity with one way functions. After authenticating a potential covert partner, participating peers send data over an encrypted covert channel using one a standard BitTorrent message type. We have also SURREAL’s security robustness to potential attacks. Finally, we have validated SURREAL’s performance and presented results comparing it with [4] and [5].

Keywords—Authentication, BitTorrent, covert channel, handshake, information hiding, p2p networks, security, steganography.

I. INTRODUCTION

With the rapid evolution and widespread adoption of peer-to-peer (p2p) networks, for information sharing, preventing sharing of both legitimate and illegitimate information – such as music, movies, software, and child pornography – even on p2p overt channels is a daunting task. Furthermore, given that, even the detection of covert channel (CC) is near impossible, preventing illegitimate CC information sharing is simply impossible. p2p networks contain ad-hoc, decentralized structures and autonomous peers who can join and leave the network at will. These characteristics of the p2p networks provide a fertile ground for malicious actors. Therefore, security requirements for p2p networks are significantly more challenging.

Steganography, a very old, popular and powerful CC technique, has been applied in numerous domains with numerous carrier file types – such as audio, video, and graphics. Contemporary steganography methods that implement CC within network traffic are highly dependent on the particular cover file data or the network protocol. While there are numerous variations of steganography implementations, most of them use the same fundamental information hiding principles.

p2p networks, by default, offers peer anonymity – a critical requirement for the adversary in scenarios such as distributing malware, child pornography, miscellaneous contraband, or when deploying botnets – to prevent attack-source traceback. p2p networks also offer network size anonymity, with which the extent of the network is never truly known, leaving a window of opportunity for the attackers to find alternate routes, attacks, and attack vectors if some parts are disabled. Therefore, p2p networks are very attractive to cybercriminals. However, p2p BitTorrent (BT) protocol’s standard 3-step handshake lacks authentication services – a key requirement in covert communications to protect the confidentiality of information and anonymity of the peers.

Therefore, in this paper, we propose SURREAL – a mechanism for establishing authenticated CC over p2p BitTorrent (BT) protocol. SURREAL implements covert peer communication building on steganography-based information hiding. In SURREAL, the covert message is first encrypted with the sender’s choice of encryption algorithm. Subsequently, the encrypted secret message is hidden in a CC within the standard BT overt communication channel – one that is overt and open to all sorts of monitoring and sniffing. Hence, security and robustness against detection and confidentiality breaches are very critical.

We have implemented a proof-of-concept prototype of the proposed SURREAL using SNARK, an open-source java framework of the BT protocol. However, one key requirement for the proposed SURREAL to survive detection is that the CCs must be built on ubiquitous protocols such that the cover communication blends inseparably with legitimate network traffic, yet, the protocol obscures secret messages such that they are not retrievable by a non-colluding peer.

A. Background and Preliminaries

Covert Channel (CC) is a communication channel, that utilizes a portion of the main channel’s bandwidth to transport secondary information, with or without malicious intentions. However, CC communication is a violation of security of the underlying communication network.

The communication through CC continues to be a significant challenge, especially since even its existence is almost
always unknown. CCs can be used for legitimate and malicious purposes — censorship violations, anonymity and privacy preservation, profiling users for business interests, national security matters, etc.

**BitTorrent (BT) Protocol.** A p2p protocol, in which “tracker” server keeps track of file copies residing on peer machines, availability when clients’ request, and also helps to coordinate efficient transmission and reassembly. The peer wire protocol consists of an initial \(\langle\text{handshake}\rangle\). After that, peers communicate via an exchange of integer length-prefixed messages. The \(\langle\text{handshake}\rangle\) is a required message, of size \([49 + \text{len(pstr)}]\) bytes, must be the first message transmitted by the client, and has the format \(\langle\text{handshake}\rangle\).

A peer message exchange starts once a client has selected a torrent file. The peers start with a \(\langle\text{handshake}\rangle\) message followed by a \(\langle\text{bit_field}\rangle\) message, which has the fixed ID=5. The \(\langle\text{bit_field}\rangle\) message may only be sent immediately after the handshaking sequence is completed, and before any other messages are sent. It is optional, and need not be sent if a client has no pieces. This is followed by other messages of one of 9 types \(\langle\text{request}\rangle\) with ID=6, \(\langle\text{piece}\rangle\), with ID=7, etc. The other control messages that are intermixed for purposes of communication control include — \(\langle\text{choke}\rangle\), \(\langle\text{unchoke}\rangle\), \(\langle\text{interested}\rangle\), \(\langle\text{not_interested}\rangle\), and cancel [12].

In the research presented in this paper, we will primarily focus on the following three message types — \(\langle\text{handshake}\rangle\), \(\langle\text{bit_field}\rangle\), and \(\langle\text{piece}\rangle\). In addition to \(\langle\text{handshake}\rangle\), there are several other message types all of which take the form \(\langle\text{length_prefix}\rangle\langle\text{message_ID}\rangle\langle\text{payload}\rangle\), where length prefix is a four byte big-endian value, message ID is a single decimal byte, and the payload is message dependent.

**Standard BT peer handshake** is a 3-step handshake and does not provide authentication services — a key requirement for covert communication. Consequently, a peer \(p_x\) has no way of knowing the identity of a connecting peer \(p_y\), except through the \(\langle\text{handshake}\rangle\) message, which can be easily spoofed. To counter this weakness, and provision authentication services, we propose design SURREAL — a 6-step authenticated covert handshake protocol for \(\langle\text{handshake}_{cov}\rangle\) message exchange containing hashed \(p_{id}\). Hence, with SURREAL, a peer shall be capable of managing its existing peer connections without solely relying on the \(p_{id}\) alone.

**B. Summary of Contributions and Roadmap**

To the best of our knowledge, SURREAL is the first attempt to present a CC by substantially enhancing the standard BT protocol. SURREAL is the first attempt at a synchronous CC that modifies the \(\langle\text{handshake}\rangle\), \(\langle\text{bit_field}\rangle\), and the \(\langle\text{piece}\rangle\) messages of a standard BT. SURREAL enhances the standard 3-step BT handshake into a 6-step authenticated covert handshake \(\langle\text{handshake}_{cov}\rangle\) enabling not only peer authentication service, but also providing robust peer anonymity service, in addition to confidentiality and data integrity services.

SURREAL is secure and robust to popular p2p attacks and colluding peers can use any arbitrary torrent for covert message exchange. We have discussed its security robustness to the three core security requirements — Confidentiality, Integrity (Authentication), and Availability. Finally, we have compared the performance of SURREAL with two other state-of-the-art techniques [4] and [5] and confirm that it achieves significantly higher throughput per round of protocol execution.

The rest of the paper is organized as follows. In section II, we discuss SURREAL system model followed by the details of SURREAL design in section III. Then, in section IV-A, we discuss the implementation details of SURREAL followed by its security analysis in section-V. Finally, in section VI, we present relevant related works followed by conclusion with directions for future research in section VII.

**II. System Model**

We assume two types of peer nodes — regular (non-colluding) peers \(p_{id}^{reg}\) and covert peers \(p_{id}^{cov}\). A given \(p_{id}\) satisfies \(p_{id}^{reg} \in p_{id}^{cov}\) or \(p_{id}^{cov} \in p_{id}^{reg}\). Each peer maintains two lists — \(\text{Lid}^{cov}\) is the list of potential covert partners to be authenticated, and \(\text{Lid}^{cov}\) is the list of authenticated covert partners.

Additionally, all covert peers have access to a pre-shared secret key, \(PSK\), which is used for message encryption/decryption, and for cryptographic operations, any symmetric key algorithm such as 3-DES or AES can be used. Also, during encryption, if \(PSK\) is shorter than \(D\), then \(PSK\) is repeated to make the two equal. To being, all potential \(p_{id}\)’s are kept in the \(\text{Lid}^{cov}\), and the authenticated ones are moved to \(\text{Lid}^{cov}\). During the \(\langle\text{handshake}_{cov}\rangle\), peers also exchange a unique mask value that is randomly generated for each communication session, to be used along with \(PSK\) for encrypting their communications.

A key aspect of the standard BT is that only two message types carry a payload data of any significant size — \(\langle\text{bit_field}\rangle\)
and \langle piece \rangle messages [3]. However, since \langle bit_field \rangle message is sent only once, it provides limited utility by constraining the CC capacity impeding the long term covert message throughput. For this reason, we choose the \langle piece \rangle message as our covert data delivery vehicle in designing SURREAL.

Finally, in designing SURREAL, we assume that the sender over the CC is also the sender over the overt channel. Hence, sender can potentially manipulate the overt channel properties to influence the characteristics of CC to maximize his payoff.

III. SURREAL DESIGN

We begin with a discussion on the proposed 6-step \langle hand_shake_con \rangle protocol, which we build on top of the existing BT’s 3-step handshake process. Then we discuss the covert communication using SURREAL, with detailed examples and the associated algorithms. In our proposed SURREAL, we consider two peers, denoted as \texttt{p}_x and \texttt{p}_y, who transfer information overtly on the network, but employ data hiding within BT protocol to communicate covert information.

A. The \langle hand_shake_con \rangle Message

The \texttt{p}_id in a standard BT \langle hand_shake \rangle message is usually the same as the \texttt{p}_id transmitted in the tracker requests. In designing SURREAL, we have modified this \langle hand_shake \rangle message to be the sha1 hash of the triple \langle \texttt{p}_id, PSK, \texttt{p}_id \rangle, as shown in equation 1.

\[
\langle \text{hand_shake_con} \rangle^{x \rightarrow y} = \text{sha1}(p_x, PSK, p_x^{mask})
\]  

(1)

This modification provides strong anonymity since sha1 is a one-way function. In this triple, the peer mask, \texttt{p}_id^{mask}, is randomly generated by each peer, which is exchanges with all the other peers on its \texttt{L}_id^con, and is used in establishing a secure covert channel. Figure 1 compares \langle hand_shake \rangle and \langle hand_shake_con \rangle.

For illustration, consider two colluding peers Alice(\texttt{p}_A) and Bob(\texttt{p}_B). To begin, \texttt{p}_A computes \langle hand_shake_con \rangle^{A \rightarrow B} (equation 2), and exchanges it with \texttt{p}_B. Upon receiving \langle hand_shake_con \rangle^{A \rightarrow B}, \texttt{p}_B is not certain if \texttt{p}_A is a covert partner, because sha1(\texttt{p}_A || PSK || \texttt{p}_A^{mask}) \notin \texttt{L}_id^con. Therefore, \texttt{p}_B saves \texttt{p}_A from the \langle hand_shake_con \rangle^{A \rightarrow B} on \texttt{L}_id^{con} for verification. \texttt{p}_B treats \texttt{p}_A as a potential covert partner and responds with \langle hand_shake_con \rangle^{B \rightarrow A} (equation 3), from which \texttt{p}_A extracts and saves \texttt{p}_B on \texttt{L}_A^{con} for verification.

It is important to note that since peers have no way to know the \texttt{p}_id of an incoming connection, except the handshake, they must treat all incoming connections as potential partners and need to authenticate them prior to engaging in covert communication. It then computes sha1(\texttt{p}_A || PSK || \texttt{p}_A^{mask}) for each \texttt{p}_id on \texttt{L}_id^{con}. Only if the \texttt{p}_id sent in the \langle hand_shake_con \rangle matches the sha1 value for a \texttt{p}_id (\texttt{p}_A in this case) on \texttt{L}_id^{con}, \texttt{p}_A will be moved to \texttt{L}_id^{con} and receives \langle bit_field \rangle (equation 5) from \texttt{p}_B. Figure 1-A compares the format of standard BT \langle bit_field \rangle with SURREAL.

\[
\langle \text{hand_shake_con} \rangle^{A \rightarrow B} = \text{sha1}(p_A || PSK || p_A^{mask})
\]  

(2)

\[
\langle \text{hand_shake_con} \rangle^{B \rightarrow A} = \text{sha1}(p_B || PSK || p_B^{mask})
\]  

(3)

B. The \langle bit_field \rangle Message

In the standard BT protocol execution, the peers exchange a \langle bit_field \rangle message immediately after the \langle hand_shake \rangle. The \langle bit_field \rangle message has the following format: \langle len=0001 + X \rangle \langle id = 5 \rangle \langle bit_field \rangle. The \langle bit_field \rangle message contains 1’s for pieces that the peer has, and 0’s otherwise. In SURREAL, \texttt{p}_A will compute a modified \langle bit_field \rangle message (equation 4) and exchange it with \texttt{p}_B. \texttt{p}_B then computes (PSK || \langle bit_field \rangle) to extract \texttt{p}_A^{mask}.

\[
\langle \text{bit_field} \rangle^{A \rightarrow B} = [PSK + p_A^{mask}]
\]  

(4)

\[
\langle \text{bit_field} \rangle^{B \rightarrow A} = [PSK + p_B^{mask}]
\]  

(5)

C. The \langle piece \rangle Message

The standard BT \langle piece \rangle message has the format:

\[
\langle piece \rangle = [\langle \text{len = 0009 + X} \rangle \langle \text{id = 7} \rangle \langle \text{index} \rangle \langle \text{begin} \rangle \langle \text{block} \rangle]
\]  

(6)

Algorithm 2 Message Encryption in SURREAL

1: \textbf{procedure} \textit{Encryption}  
2: \texttt{p}_x, \texttt{p}_y : \textit{colluding_peers}  
3: \texttt{p}_x \leftarrow \textit{sending_peer}; \texttt{p}_y \leftarrow \textit{receiving_peer}  
4: if \texttt{p}_x has \langle (S) \& \& (piece) \rangle in \langle \text{block}_D \rangle then  
5: \texttt{D}_x \leftarrow \texttt{D} + \texttt{PSK}  
6: if \texttt{M}[\text{length} \langle \text{M} \rangle] = 1 then  
7: \texttt{S}_y \leftarrow \textit{pad}(\texttt{S})  
8: \textbf{for all} \texttt{[n]} \textbf{do}  
9: if \texttt{M}[\text{length} \langle \text{M} \rangle] = 1 then  
10: \texttt{D}[\texttt{n}] \leftarrow \texttt{D}[\texttt{n}] + \texttt{S}_y[\texttt{n} + 1]  
11: \textbf{else} \texttt{D}[\texttt{n}] \leftarrow \textit{rand}(0, 1)  
12: \textbf{Return} \texttt{D}'

Algorithm 3 Message Decryption in SURREAL

1: \textbf{procedure} \textit{Decryption}  
2: \texttt{input} \leftarrow \langle \texttt{D}', \texttt{D}, \texttt{PSK}, \texttt{M} \rangle  
3: \texttt{D}'_x \leftarrow \texttt{D}' + \texttt{D}  
4: \texttt{D}_x \leftarrow \texttt{D}'_x + \texttt{PSK}  
5: \texttt{count} \leftarrow 0  
6: \textbf{for all} \texttt{[n]} \textbf{do}  
7: if \texttt{M}[\text{length} \langle \text{M} \rangle] = 1 then  
8: \texttt{S}_y[\texttt{count}] \leftarrow \texttt{D}_x[\texttt{n}]  
9: \texttt{count} \leftarrow [\texttt{count} + 1]  
10: \textbf{return} \texttt{S}
where \langle index \rangle specifies the zero-based piece index, \langle begin \rangle specifies the zero-based byte offset within the piece, \langle block \rangle is the actual data, which is a subset of the piece specified by index. In SURREAL, \( p_A \) sends a \langle piece_{cov} \rangle message, with the sha1 hash of \( p_B^{mask} \) as the payload.

\[
\langle \text{piece} \rangle^{A \rightarrow B} = [\text{index} = 0; \text{begin} = 0; \text{block} = \text{sha}1(p_B^{mask})]
\]

(7)

\[
\langle \text{piece} \rangle^{B \rightarrow A} = [\text{index} = 0; \text{begin} = 0; \text{block} = \text{sha}1(p_A^{mask})]
\]

(8)

The message should indicate that for this \langle piece_{cov} \rangle, index = 0, begin = 0, and block = \( p_A^{mask} \) (equation 7), \( p_B \) compares sha1 hash from the \langle piece_{cov} \rangle with the true sha1 hash of \( p_B^{mask} \). A successful match authenticates the identity of \( p_A \) to \( p_B \). Similarly, when \( p_B \) sends \langle piece_{cov} \rangle with the sha1 hash of \( p_A^{mask} \) as the payload (equation 8), \( p_A \) authenticates \( p_B \) as above, prior to covert communication. Once \( p_A \) and \( p_B \) have successfully authenticated each other, it completes a two-way authenticated CC between \( p_A \) and \( p_B \). Figure 3 compares the \langle piece \rangle with \langle piece_{cov} \rangle.

D. SURREAL Operations – Discussions

During the initial \langle hand_shake \rangle and \langle bit_field \rangle exchange, three specific situations will result in peer severing the connection – 1) connecting peers do not receive a matching info hash, 2) requesting peer does not receive the expected \( p_id \), and 3) \langle bit_field \rangle message is longer than expected or has reserved bits overwritten [12]. Once peers get past this stage, then communications are controlled through peer messages such as choke or interested messages.

Message En/Decryption with the random mask. For discussions in this section, let us assume \( p_x \), \( p_y \) are colluding peers. In figure 4, let \( S \) be the secret message, \( M \) be the mask of \( p_x \), and \( D \) be the next block to be transmitted. Algorithm 2 and figure 4(a) collectively illustrate SURREAL's encryption of \langle block \rangle portion of the \langle piece_{cov} \rangle message prior to transmission.

Algorithm 1 is the routine that is used during encryption to check for significant bits to hold the covert message. Note that the \( M \) is used to determine which bits of the data payload contain hidden data and \( M \)'s length is equal to the number of pieces in the torrent being exchanged by the swarm. First, \( p_x \) will compute \( D_x = [D \oplus PSK] \). Then, using \( M \), \( p_x \) will compute \( M(D_x) \) - the significant bits of \( D \). Now, if needed, \( p_x \) will pad \( S \) starting with '1' followed by required number of '0's to make \( S.length = M(D_x).length \), and denote the padded version as \( S_p \). Finally, \( p_x \) will compute \( D' \), the encrypted secret that it transmits to \( p_y \). The encryption process is presented with an example, capturing the idea intuitively, in Algorithm 2 and figure 4(a).

\( p_y \), upon receiving \( D' \), will retain all \langle piece_{cov} \rangle messages and decrypts once all relevant \langle piece_{cov} \rangle have been received. In order to successfully decrypt \( D' \) to obtain \( S \), \( p_y \) must know the PSK and \( M \) used for encryption. Additionally, the peer must also have a valid copy of \( D \), which was used to generate \( D' \). A valid copy of \( D \) may be obtained from either \( p_x^{reg} \) or \( p_x^{cov} \) in the BT swarm during the course of normal BT activities. Algorithm 3 and figure 4(b) graphically depict the decryption process.

An important contributor to the strength of this method is the randomly generated one-time mask used to determine the significant bits – data payload bits that will actually contain the hidden data. Because it is transmitted using the \langle bit_field \rangle message during the \langle hand_shake \rangle protocol, its length in bits will be equal to the number of pieces in the torrent being exchanged by the swarm. Therefore, it is better to use a meta-info file that describes a large set of data when creating the CC, because such a meta-info file will generate a longer mask than one describing a small set of data.

In equation 6, \langle block \rangle is a portion of the data being shared via the torrent, \langle index \rangle specifies the source \langle piece \rangle for the data in the \langle block \rangle, and \langle begin \rangle specifies the corresponding \langle block \)'s offset from the beginning of the \langle piece \rangle. In equation 9, \( \text{ones}(\mathcal{M}) \) represents the number of bits in \( M \) set to 1; and \( M[n] = \mathcal{M}[\text{D.size mod } M\text{.size}] \).

\[
\mathcal{E} = \frac{\text{ones}(\mathcal{M})}{2(\text{len}(D))} \times \left\lfloor \frac{\text{len}(D)}{\text{len}(\mathcal{M})} \right\rfloor + \text{ones}(\mathcal{M}[0],\mathcal{M}[1],\ldots,\mathcal{M}[n])
\]

(9)

The efficiency \( \mathcal{E} \) of SURREAL’s CC, as shown in equation 9, is the ratio of the number of significant bits for a given \( \langle M, D \rangle \) to \( 2 \times D.length \), since the original \( D \) must also be transmitted in order for the receiving peer to be able to decrypt the received message. However, obtaining the original message is simple during regular BT operations as previously noted. With a randomly generated mask, the average case efficiency \( \mathcal{E} \) of the CC will be approximately 0.25, i.e., 2 bits of covert data per byte of \( D \).

IV. EXPERIMENTS AND PERFORMANCE VALIDATION

A. SURREAL Implementation

The original BT protocol is simple and flexible enough to incorporate the changes required for this proof of concept. The key requirement for extending the standard handshake into an extended covert handshake is for the client to appear to be operating normally to an outside observer. We chose the open-source Snark Project as our foundation for implementing the BT CC. Snark is an open-source client for downloading and sharing BT files.

Although Snark can be used as a regular BT client, it was mainly developed to explore the BT protocol, and therefore it is a simple implementation that follows the published BT specification. The Snark Project itself is developed in java with GNU Compiler for Java (gcc) [12]. A key reason for choosing Snark is that, because it is an older implementation of a BT client, it follows the published BT standard, without any modifications or extensions.

The original BT protocol is simple and flexible enough to incorporate the changes required for our proof of concept. While the Snark program is designed as both a standard BT client and server for sharing local files, we once again focus on the handshake, bitfield, and the piece messages for our proof-of-concept implementation of SURREAL.
B. SURREAL Comparison with other State-of-the-art

Desimone et al. [5] have the client send covert messages by hiding them in the $p_{id}$ field in an ⟨ announce ⟩ request. The sender connects to the tracker’s web interface and sends an announce request with a $p_{id}$ containing a covert message. This message is then stored in the tracker’s database.

To retrieve this message, the receiver performs an announce request to the tracker as a legitimate p2p client. The server replies to the receiver with a list of clients active in the specified torrent. This reply will contain the covert message. In this CC, 20bytes of information can be sent at a time. However, for a more legitimate-looking CC, they recommend reducing it to 12bytes. Unlike this, SURREAL uses the variable length ⟨ piece ⟩ message, with which clients can send significantly larger amounts of encrypted covert data.

Cunche et al. [4] propose an asynchronous CC mechanism based on BT trackers that uses HTTP commands. There is no direct message exchange between the covert peers. Instead, a centralized BT tracker is utilized for storing covert messages. They have analyzed the detectability of CCs by an adversary and show that its detection is unlikely. However, this mechanism appears to have some key limitations. Despite being asynchronous, it has limited recovery window by the receiving peer. Also, CC capacity is very limited – 122 bits per announce request. Most importantly, since none of the colluding peers have supervisory/administrative control of the tracker server, their mechanism completely hinges on the fact that the tracker’s behavior and configuration will remain consistent. In a real world scenario, this is rather a very strong assumption. Covert communication among peers should never be coupled with a system that is not in direct control of the colluding peers. In SURREAL, this is overcome with client side implementation.

C. Experiments and Results

We have compared SURREAL with [4] and [5], as discussed in the previous section. In figure 5, we have presented results comparing the average number of messages required to transmit covert messages of varying sizes between 5KB and 100MB. The results compare number of ⟨ piece_con ⟩ messages that surreal needs, with an efficiency $E = 25\%$, to completely transfer covert messages of the above sizes, with that of [4] with 12bytes of covert data per announce message and [5] with 122bits of covert data per request.

V. SURREAL – SECURITY ANALYSIS

We now analyze SURREAL’s security robustness to common and powerful attacks targeted at p2p networks.

A. Integrity Attacks and Defense

Replay attacks. SURREAL mandates that each peer reply with a sha1 of its partner’s mask to complete the authenticated ⟨ hand_shake_con ⟩. The hash value of the mask is intended to provide confidentiality, making the mask robust to packet sniffing and subsequently brute forcing. Additionally, the freshness of the randomly generated session mask can readily thwart packet replay attacks.

Impersonation attacks. Peer $p_x$ can impersonate another peer $p_y$ initially by simply replaying a previous ⟨ hand_shake ⟩ and ⟨ bit_field ⟩ message sent by $p_y$ to a third peer $p_z$. However, when $p_z$ challenges $p_x$ to decrypt $p_{mask}^z$, $p_x$ will be unsuccessful since the mask is different from what $p_x$ replayed. Consequently, the attacker – $p_x$ in this case – will not be able to reply with a valid ⟨ piece ⟩ message to finish the authenticated ⟨ hand_shake ⟩ with peer $p_z$.

Modification attacks. Messages subsequent to the initial ⟨ hand_shake ⟩ are encrypted. A MITM attacker will not be able to decrypt the messages without the decryption key. Even if an attacker knows the $PSK$ and the $p_{id}$ it intends to attack, the $p_{mask}$ value is unique and randomly generated for each session. Hence, the adversary cannot successfully modify the contents of messages exchanged.

VI. RELATED WORK

With the evolution of of p2p networks, there has been an increasing interest, especially in the BT protocol. Numerous works have explored the use of covert channels in botnets in numerous contexts including – malware distribution [9]; DNS-based command and control [1]; and torrent files-based storage CC [11]. Numerous CC techniques have also been proposed for different network and application protocols.
In [6], Ahsan and Kundur’s present a CC technique that manipulates IPv4 header and packet ordering in an IPSec environment to pass supplementary information. In [7], Eidenbenz et al. demonstrate how BT-like protocols can be exploited to accomplish steganography handshake and broadcast in point-to-point networks. Their methods do not provide security guarantees of authentication, confidentiality, and anonymity.

In [2] authors present a technique to exploit Vuze, a popular file-sharing client. One of the advantages of Vuze is its improved efficiency implemented through a network coordinate system. However, network coordinate systems are inherently insecure and a malicious peer can lie about its coordinate to appear closest to every peer in the network consequently hijacking every search query.

Goudar et al., in [8] propose a system that uses features of both cryptography and steganography, with a TCP/IP header serving as a steganographic carrier to hide encrypted data. Yaroshkin et al. [13] have presented a detailed discussion on covert communication in p2p architecture. While there solutions have good methods for establishing covert communication, they do not provide the most important requirement - peer authentication.

In [11], Li et al. propose “Stego-Torrent” for torrent files which uses two approaches to hide data – 1) letters in the URL are changed according to the secret message, and 2) some optional fields in the torrent file are reused. On the other hand, “StegTorrent” [10] is another steganography scheme developed over BT. In this approach, the order of data packets in the exchange protocol is changed to create the CC. The basis for development of this scheme is one-to-many packet transmissions in the Torrent protocol, with the header support for packet numbering and retrieval of packet sequence. Neither of the above two methods provide peer authentication, a key requirement for covert communication.

VII. Conclusion and Future Work

In this paper, we have presented SURREAL, which is an extension of the standard BT p2p protocol. SURREAL extends BT’s standard 3-step \( \langle \text{hand\_shake} \rangle \) to a 6-step \( \langle \text{hand\_shake}_{\text{con}} \rangle \) secure peer authentication service and robust peer anonymity with a one-way function.

In designing SURREAL, we primarily focus on three standard BT message types extending them to support CC – \( \langle \text{hand\_shake}_{\text{con}} \rangle \), \( \langle \text{bit\_field}_{\text{con}} \rangle \), and \( \langle \text{piece}_{\text{con}} \rangle \). We have implemented SURREAL’s proof-of-concept using the open-source Snark, and validated its performance by comparing it with [4] and [5]. The results confirm that SURREAL outperforms [4] and [5] by a significant margin.

REFERENCES