

Did You See That? A Covert Channel Exploiting Recent Legitimate Traffic

Steffen Wendzel *Member, IEEE*, Tobias Schmidbauer, Sebastian Zillien, Jörg Keller

Abstract—Covert channels are unforeseen and stealthy communication channels that enable manifold adversary scenarios, such as the covert exfiltration of confidential data or the stealthy orchestration of botnets. However, they can also allow the exchange of confidential information by journalists. All covert channels described until now therefore need to craft seemingly legitimate information flows for their information exchange, mimicking unsuspicious behavior.

In this paper, we present DYST (*Did You See That?*), which represents a new class of covert channels we call *history covert channels*. History covert channels can communicate *almost exclusively* based on *unaltered legitimate* traffic created by regular nodes participating in a network. Only a negligible fraction of the covert communication process requires the transfer of actual covert channel information. We extend the current taxonomy for covert channels to show how history channels can be categorized.

We theoretically analyze the characteristics of history channels and show how their configuration can be optimized for two channel implementations, called DYST-Basic and DYST-Ext. We further implement a proof-of-concept code for both DYST variants and evaluate the performance (robustness, detectability, and optimization) with both, simulated and real traffic. Finally, we discuss application scenarios and potential countermeasures against DYST.

Index Terms—Covert Channel, Steganography, Information Hiding, Network Security, Anomaly Detection.

I. INTRODUCTION

COVERT channels are policy-breaking and stealthy communication channels that are not foreseen in a system's design [1], [2]. Such channels are regularly used to transfer secret information, e.g., for the purpose of data exfiltration or malware communications [3], [4]. However, covert channels can also be applied for censorship circumvention, e.g., by journalists [3], [5]. Covert channels have been investigated for different environments, including networks [3], [6], [7], cyber-physical systems [8]–[11], local processes/systems [12]–[14] and in out-of-band scenarios, such as ultrasonic sound, light, vibration, radio-frequency, magnetic fields or temperature [4], [15].

In this work, we focus on *network-based* covert channels (however, the proposed approach can be adopted in various scenarios, not limited to network environments). All known network covert channels must *embed* secret messages into network communications. Since their first appearance 50 years

ago by B. W. Lampson [1], authors have performed one of two required actions: (1) they either relied on the creation of *own* traffic (so-called *active* sending), in which they embed the secret data; or (2) they *modify* legitimate traffic transmitted by third-party nodes to embed the secret data (called *passive* sending). In both cases, the embedding of secret information renders a channel slightly detectable. This aspect is the central limitation of all previously known covert channels.

To overcome this limitation, it would need a covert channel that relies (at least largely) on *unmodified* legitimate traffic. In this paper, we present a new covert channel called DYST (*Did You See That?*) that fulfills this criterion for the first time, since Lampson founded this research area in 1973.

In particular, our contributions are as follows:

- 1) We introduce a novel threat in the form of a new class of covert channels that we call *history covert channels*. These covert channels advance over state-of-the-art as follows: 1) they generate only minimal signaling traffic and *no* data traffic; 2) they do not rely on *modifying* existing traffic to transfer a secret message; 3) by sending only one or few covert signaling bits of information, they can transfer several secret data bits. These features render history channels upmost challenging to detect.
- 2) We extend the existing taxonomy for network covert channels with a new category called *fully-passive sending* to reflect all components of this new class of covert channels.
- 3) We provide the first implementation of a history covert channel, called DYST to confirm the feasibility of such channels in practice. DYST contains a *data* channel that requires no own or modified traffic and a *signal* channel that consists solely of rarely sent ARP broadcast messages with requests for legitimate systems (representing only 1 covert bit).
- 4) As history covert channels comprise a whole family of variants, we provide a theoretical analysis of their performance and optimization.
- 5) We evaluate DYST's robustness, detectability, and optimization under different settings and show that DYST allows the transfer of variable secret data bits through only 1 covert signaling bit, which state-of-the-art covert channels do not achieve.

The remainder of this paper is structured as follows. Sect. II provides background information and discusses related work while Sect. III presents the functioning as well as the theoretical description and optimization of DYST. We describe our experimental testbed and the evaluation of DYST in Sect. IV.

S. Wendzel and S. Zillien are with the Center for Technology and Transfer (ZTT), Hochschule Worms, Germany

E-mail: {wendzel,szillien}@hs-worms.de

S. Wendzel, T. Schmidbauer and J. Keller are (also) with the Faculty of Mathematics & Computer Science at the FernUniversität in Hagen, Germany
E-mail: tobias.schmidbauer@studium.fernuni-hagen.de, joerg.keller@fernuni-hagen.de

A discussion is provided through Sect. V. Finally, Sect. VI concludes and gives an outlook on future work.

II. BACKGROUND & RELATED WORK

Covert Channels: A *Covert Channel* exchanges information in a stealthy manner between a *covert sender* (CS) and one or more *covert receiver(s)* (CR). A covert channel is one that is not foreseen in a system's design [1] and relies on the concept of policy-breaking communication [2]. If the covert information can be received by more than one CR, the communication can be considered as a multicast or even broadcast covert channel. In computer networks, the covert channel nests into a network protocol, e.g., by manipulating bits of a packet header or by adjusting the delays between successive network packets.

In [16], Cabuk et al. propose the idea of an advanced version of a covert channel based on delays between packets: the covert channel transmits the hidden data by modulating the delays between consecutive network packets. The advanced version of their covert channel mixes covert transmissions with sections of real, legitimate network traffic. This helps to skew the statistics and makes the detection of the covert channel harder. The difference to our approach is that Cabuk et al. use sections of legitimate traffic solely introduce noise into the actual covert channel signal. The legitimate traffic carries no hidden information at all.

Several additional methods work similarly to the one of Cabuk et al. For instance, *JitterBug* by Groza et al. [17] adds random delays to legitimate Telnet traffic. Walls et al. proposed *Liquid* [18], an extension of *JitterBug*, in which they split the channel into “transmitting” and “shaping” delays (shaping delays carry no information but manipulate the statistics of traffic). Similarly, Gianvecchio et al. [19] tailor traffic automatically based on the statistical characteristics of legitimate traffic. Again for all these approaches, artificial modifications are performed to transfer secret information, even if based on legitimate traffic, which is the key difference to DYST.

There are also approaches that work on a more abstract level. Yarochkin et al. [20] proposed the so-called *network environment learning* phase. This approach was used solely to determine which protocols occur regularly in a network to succeedingly exploit only these protocols for covert communication. No work is known that exploits legitimate traffic for a covert channel. Moreover, the covert channel of Yarochkin et al. did not split the covert channel's control channel from its data channel as we do.

Image steganography uses a variant called cover selection, where a database of images is used, a hash function is applied to each image, and if the secret message matches the hash value, the image is sent by the covert sender [21]. In contrast, our method relies on network traffic that is transmitted anyway, and only uses signals such as extra ARP requests, instead of generating extra traffic such as images, even if those images are innocent.

There is one method that actually splits the control channel from the data channel: as shown by Wendzel and Keller in [22], several covert channels propose to utilize internal

control protocols. Therefore, a covert channel is nested into the utilizable bit areas of a network packet. Some of the utilizable bits are used for the control protocol while others are used for the data channel. However, that approach has a major limitation compared to ours as both, the control and the data channel, reside in the same packet and modify legitimate packets or craft new packets instead of exploiting solely unaltered legitimate traffic for the data channel.

Finally, there is one proposal by Caviglione et al. from botnet research that works by waiting for a pre-defined network packet sequence [23]. If the sequence occurs, all botnet nodes would perform a certain action. The idea was solely described on a conceptual level and was not implemented by the authors. Further, their concept did not involve the option to influence which secret message is transferred, which makes it fundamentally different from our history channels.

Network Covert Channel Detection: Several detection methods for covert channels have been proposed throughout the years. Popular ones are, e.g., compressibility score [24], ϵ -similarity [25], regularity metric [25], a method from Berk et al. [26], as well as classical methods, such as Kullback-Leibler divergence test [27], Kolmogorov-Smirnov [27] or entropy-based analyses [7]. All of these methods require at least a few hundred covert channel packets to provide somehow reliable detection of covert channel flows. In contrast, history channels send few *signaling* packets per time, resulting in only a minimal influence on a flow while the data flow of history channels is entirely legitimate and thus indistinguishable. We evaluate two common detection methods on DYST in Sect. IV.

III. THE HISTORY COVERT CHANNEL METHOD

In this section, we first discuss the requirements of our history covert channel, followed by a description of the detailed functionality of DYST. Further, we explain the chosen parameters for DYST, including the optimization, and finally extend the existing taxonomy of active and passive covert channels.

Definition 1. A **history covert channel** is one that *points* to already existing (live or stored) data that matches a secret message instead of *sending* a secret message itself. The only covertly transferred information is the pointer. ■

Note. We chose the specific PoC implementation discussed in the remainder for reasons to simplify experiments and explanations. We note, however, that history covert channels are not restricted to local networks and more variants are possible, cf. Sect. V.

A. Requirements

DYST relies on some crucial characteristics of the environment in which CS and CR operate:

- 1) CS and CR must be able to observe some messages which they both receive (almost) at the same time. There are different options to achieve that: *i)* CS and CR could read broadcast messages in a local WiFi network that both of them receive; *ii)* CS and CR could only evaluate messages that pass through the routing path of CS and CR (e.g., when both act as routers); *iii)* CS and CR

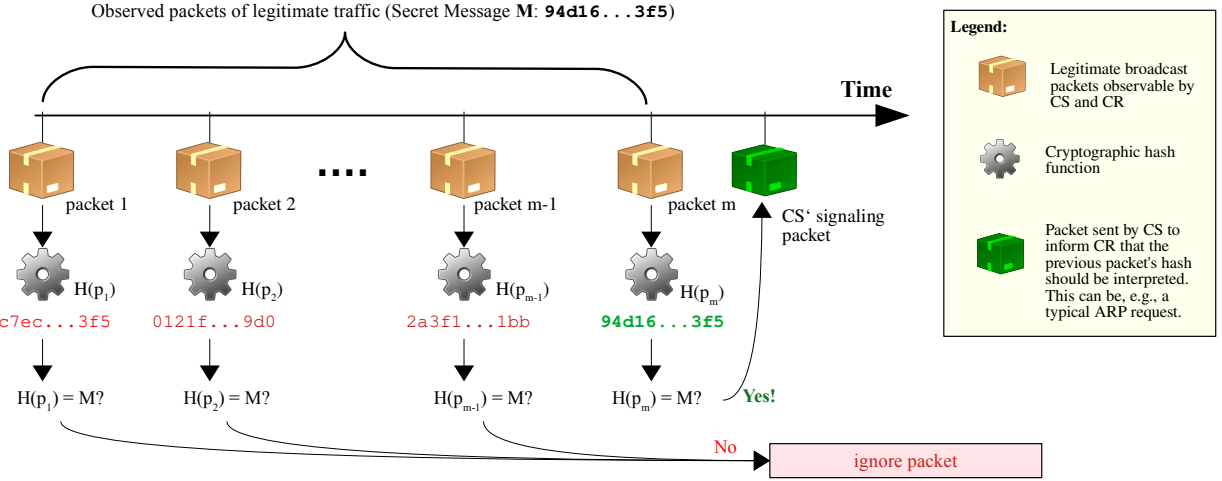


Fig. 1. The DYST-Basic Sending Process.

are members of a multicast-group, receiving frequent updates (for instance IGMP, like exploited for an active indirect covert channel in [28]);

- 2) CS and CR need to utilize the same hashing function and the same input parameters (i.e., which packets and which header and body fields of these packets, timestamps, and so forth) to create similar data;
- 3) Moreover, CS must be able to signal CR some information in a timely manner, e.g., CS might be able to send a legitimate ARP broadcast to the local WiFi network, which is also observable by CR.

Optionally, CS and CR may even reside on two existing legitimate systems as there is no need for CS and CR to utilize added systems. The above-mentioned conditions can also be fulfilled outside the scope of networks.

B. Functioning of DYST

We realized two variants of DYST: first, we describe DYST-Basic, a simple variant where all bits of an observed hash need to match the chunk of the covert message. Second, we will describe DYST-Ext, which is an extended variant capable of transferring correct information, though the hashes are not matching perfectly. This approach creates more variants to choose from in the tradeoff between steganographic bandwidth and detectability.

1) *DYST-Basic*: For DYST-Basic, the following steps must be performed continuously until a whole message is transferred (see Fig. 1):

- 1) CS and CR both record legitimate traffic with their network interfaces connected to a shared network.
- 2) For each packet p_i that both CS and CR can observe (e.g., broadcast messages), they apply a hash function $H()$ to the input values they have agreed on, to generate a hash value $h_i = H(p_i)$ of length h .

CS and CR can exchange a secret message in a bi-directional manner. In the remainder, we describe the sending process from CS to CR. However, CR can simultaneously operate as a CS and CS as a CR to send/receive data.

The sending process can be described as follows (see Fig. 1) and requires that each secret message M to be sent has the length h . To transfer a secret message of length $k \cdot h, k \in \mathbb{N}, k \geq 2$, the message must be split into k fragments and the sending process must be performed successively for each of the k fragments.

- 1) To signal a secret message of length h , CS waits for a packet p_i with a hash value $h_i = H(p_i)$ which equals M .
- 2) After the CS observed such a packet, it sends a signal, for example a legitimate ARP request in which it asks for the address of some legitimate node (e.g., a router) in the network.

Finally, the receiving process is conducted as follows:

- 1) CR interprets the occurrence of this ARP broadcast from CS as a secret signal that the expected message can be observed in the data channel.
- 2) CR interprets the previous hash value that represents the covert message.

Obviously, the channel is noisy and requires error detecting (and correcting) bits or mechanisms to ensure a robust transmission of the correct information.

The major advantage of this sending procedure is that CS needs to send only one bit of covert information (represented by the ARP request) to transfer h bits of secret content. The ARP request can be replaced by any other seemingly legitimate unicast, broadcast, multicast, or anycast message, observable by the CR.

Moreover, CS reaches multiple CR simultaneously, if desired, rendering the channel a multicast or broadcast covert channel.

Assuming that each of the 2^h possible hash values is equally likely, then on average an exact match between secret message M and hash value h_i will be achieved after 2^h packets. If an exact match is not required but up to t bits can be wrong, i.e. the hamming distance between secret message M and hash value h_i can be at most t , then the chance of such a partial match would increase by a factor $\sum_{i=0}^t \binom{h}{i}$. To enable

the CR to still decode the message correctly, the message would have to be encoded with an error-correction code that allows correction of up to t bit errors. This means that only $h - c$ bits are available for the secret message itself and the remaining c bits are used for the error-correction bits [29]. For a binary block code, we have $c > 2t$ as $c = 2t$ is a bound achieved by *maximum distance separable* (MDS) codes, which however only exist in trivial form for binary¹ codes [30, Prop. 9.2]. Furthermore, besides a strong restriction on the number of message bits (the larger c , the smaller $h - c$), only some combinations of h , c , and t are available for applicable code families such as binary *Bose-Chaudhuri-Hocquenghem* (BCH) code, see e.g. [29, App. A]. Our initial investigations revealed that error-correction codes only in some cases match the performance (in terms of bandwidth and average signal distance) of DYST-Basic, and mostly perform worse. Yet we carried over the idea of using partial matches to using checksums instead of error-correction codes, creating an extended version of DYST.

2) *DYST-Ext*: The functionality of DYST-Ext works similarly to DYST-Basic, but the secret message M now only comprises $h - c$ bits, and is concatenated by CS with a c -bit checksum to an encoded message \tilde{M} of length h . When comparing \tilde{M} with h_i , CS allows up to t non-matching bits. Thus, the advantage of DYST-Ext comes from the fact that it can utilize a larger fraction of observed messages, leading to a shorter waiting time for fitting packets.

CR, upon receiving a signal (such as an ARP request) indicating a secret message transfer, again picks up the latest hash value h_i from the hash database. It then tries out all possibilities to flip up to t bits in h_i , until the checksum of the first $h - c$ bits in the modified h_i matches the last c bits in modified h_i (called a *hit*) for the first time. CR accepts the first $h - c$ bits as secret message M .

CS knows the order in which CR will apply modifications to the hash value h_i . Thus, CS can check if the first hit really will produce the message M . CS will only send an ARP request as a signal for CR if at most t bits of h_i and \tilde{M} do not match **and** the first hit found by CR will produce M . So, not all t bits matching packets can be utilized for this approach.

Fig. 2 illustrates the working of DYST-Ext.

Formally, if $C : \{0,1\}^* \rightarrow \{0,1\}^c$ is the checksum function, and $d(x, y)$ is the hamming distance between two bitvectors x and y of equal length, CS first checks if

$$d(M || C(M), h_i) \leq t. \quad (1)$$

If so, CS enumerates the set

$$S_{h_i} = \{x \in \{0,1\}^h \mid d(x, h_i) \leq t\} \quad (2)$$

in a pre-defined order, i.e., it generates a sequence of distinct bitvectors $x^{(1)}, x^{(2)}, \dots$ that together form S_{h_i} . Let $extmsg(x)$ be a function to extract the first $h - c$ bits from a bitvector of length h , while $extchksm(x)$ extracts the last c bits. For $j = 1, 2, \dots, |S_{h_i}|$, CS checks if

$$C\left(extmsg\left(x^{(j)}\right)\right) = extchksm\left(x^{(j)}\right) \quad (3)$$

¹Using non-binary codes decreases match probabilities further and thus is no option, either.

and stops with the first hit at index j^* . If

$$M = extmsg\left(x^{(j^*)}\right), \quad (4)$$

then CS sends an ARP request.

Upon receiving such an ARP request, CR looks up h_i , and also enumerates S_{h_i} according to Eq. (2), does the computations from Eq. (3) and stores secret message M according to Eq. (4).

Please note that both DYST-Basic and DYST-Ext are *families* of variants, because they are parameterized in H and h, c, t , respectively.

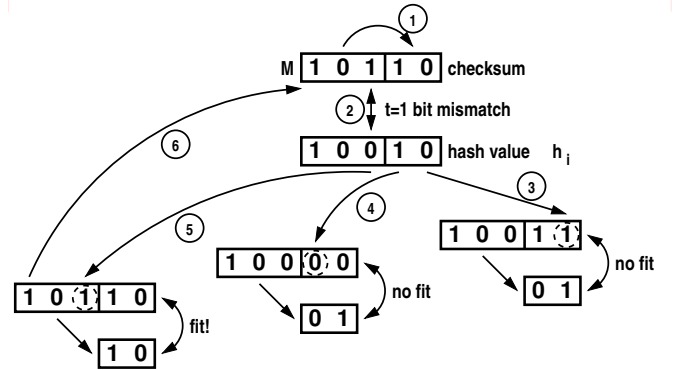


Fig. 2. Functioning of DYST-Ext. For illustration, we use a message chunk M of $h - c = 3$ bit length and a checksum of $c = 2$ bit, which simply represents the number of ones in the message chunk as a binary number (step 1). In the hash value h_i with $h = 5$ bits, only $t = 1$ bit do not match (step 2). CS thus checks all modifications of the hash value until a hit between the reconstructed message and reconstructed checksum occurs (steps 3 to 5, flipped bits marked by dashed circles, order of bit flips: right to left). As the reconstructed message equals message chunk M (step 6), CS will send an ARP request to CR. CR will perform the same computation and reconstructs M . For message chunk $M = 110$ (same checksum), the same fit would apply but not reconstruct M , and no ARP is sent.

C. Parameter Choice

1) *DYST-Basic*: We first consider the situation of DYST-Basic, where CS signals to CR if the hash value of a network packet matches the secret message exactly. If the hash function H has optimal properties, each bit of the hash value has a value of 0 or 1 with probability 50%, respectively. The probability that h_i equals a given secret message M then is 2^{-h} , as all bits of the hash value can be considered independent. As the hash values of the different packets can be considered uniformly distributed and independent, the number of packets until a match between hash value and secret message occurs follows a geometric distribution with success probability 2^{-h} , i.e., with expectation 2^h .

2) *Pareto-optimal variants*: A covert channel such as DYST-Basic can be characterized by two properties: the average *distance* between ARP signal messages (measured in the number of observed packets in between), which will influence detectability, and the *bandwidth*, i.e. the average number of secret message bits per observed packet. Thus, DYST-Basic

with parameter h is a family of covert channels with distance $dist_{basic}(h) = 2^h$ and bandwidth $bw_{basic}(h) = h/2^h$.

We would like to maximize both: signaling distances because there is a threat of detectability when ARP requests occur too often, and bandwidth to increase applicability.² Yet, increasing distance will reduce bandwidth and vice versa.

Hence, to achieve an optimal compromise between the two parameters, we search for a Pareto front, i.e., a set of non-dominated variants³. Alternatively, we impose a constraint on one parameter and search the optimal value of the other parameter, i.e., we cut the Pareto front with a vertical or horizontal line into two halves, and search the point on the front closest to the border (in the “allowable” half). It is obvious that variants of DYST-Basic with different values for h do not dominate each other, as improving one parameter makes the other worse.

3) *DYST-Ext*: In DYST-Ext, only $h - t$ or more bits of the hash value must match the encoded message, comprised of message chunk and checksum. The number of matching bits is a random variable X that is binomially distributed with h trials and success probability 0.5, and thus the probability that at most t of the h bits do not match is

$$P_h(X \geq h - t) = \sum_{j=0}^t \binom{h}{j} / 2^h. \quad (5)$$

CS and CR must try out

$$T_{h,t} = \sum_{j=0}^t \binom{h}{j} = P_h(X \geq h - t) \cdot 2^h \quad (6)$$

possible modifications of the hash value. For each, the chance that the checksum of the message part of such a modified hash value equals the checksum part of the modified hash value, i.e., the chance that Eq. (3) is fulfilled, is 2^{-c} . Thus, the number of trials until a fit will occur is geometrically distributed with parameter 2^{-c} , yet with a limited range of $T_{h,t}$ trials. The chance that the true message has the first fit thus is

$$U_{h,t,c} = \frac{1}{T_{h,t}} \cdot \sum_{k=0}^{T_{h,t}-1} (1 - 2^{-c})^k = \frac{1 - (1 - 2^{-c})^{T_{h,t}}}{T_{h,t} \cdot 2^{-c}}. \quad (7)$$

As both events (at most t non-matching bits, secret message chunk is re-constructed in first fit) can be considered independent, the chance that a message transfer can be signaled by an ARP message is their product, $P_h(X \geq h - t) \cdot U_{h,t,c}$. Taking Eq. (6) into account, we see that this product is approaching 2^{h-c} , the probability of a signal in DYST-Basic with a message of length $h - c$, yet the additional solutions can still be non-dominated. As the product probability is independent of the particular hash value, the number of packets until a signal occurs is again geometrically distributed, with the expectation

$$dist_{ext}(h, t, c) = \frac{1}{P_h(X \geq h - t) \cdot U_{h,t,c}}, \quad (8)$$

²The third parameter of the covert channel magic triangle, robustness, is considered a fixture in this optimization, as it will be considered before going into this tradeoff, cf. Section IV-D, and will reduce the number of observable packets per time so that it affects all variants in a uniform way.

³A covert channel variant A is dominated by variant B if $distance(A) \leq distance(B)$ and $bandwidth(A) \leq bandwidth(B)$.

and the covert channel has a bandwidth

$$bw_{ext}(h, t, c) = (h - c) \cdot P_h(X \geq h - c) \cdot U_{h,t,c}, \quad (9)$$

as $h - c$ bits of the secret message can be decoded by CR with each signal.

4) *Pareto Front*: We have computed distance and bandwidth for DYST-Basic with $h = 14, \dots, 20$ and for DYST-Ext with $h - c = 14, \dots, 18$, $c = 6, \dots, 10$ and $t = 1, \dots, 5$, both analytically, and supported by simulations of $5 \cdot 10^7$ hash values, where we counted how often CS signals in each variant considered. The hash values in simulations were generated by successively encrypting a 128-bit value with AES and a fixed 128-bit key, starting with value 0 and using the first h bit of the value as hash value. We used three different checksum functions in simulations (we only use first c bits of longer results): SHA-3, CRC8, and a handcrafted function that cuts the encoded message into pieces of length c , adds those pieces as binary numbers, adds 9, and takes the c lowermost bits of the result. All simulations were repeated with a second seed and results were manually compared to exclude the possibility of artifacts, which however did not show. Raw data, Pareto front data, and the simulation code are available via a repository, cf. Sect. V.

The values for h, t, c were chosen to allow comparison between different variants in a restricted range, and to illustrate development of distance and bandwidth over the range for a particular parameter.

Fig. 3(left) provides all of the above variants as points in the plane with distance on the x-axis and bandwidth on the y-axis. All points are quite close together, so we do not have a point “cloud” but still a bit “thicker” line. Thus, DYST-Ext extends DYST-Basic in the sense that the user has more choices in the tradeoff between distance (stealthiness) and bandwidth than with DYST-Basic alone. This is illustrated in Fig. 3(mid) and (right) that depict zooms into $(10^6, 0.1)$ and $(10^5, 0.001)$, respectively. The points representing variants of DYST-Ext (blue) fill the delays between the points representing variants of DYST-Basic (red). Fig. 3(right) depicts the region of interest, i.e., the region where the actual tradeoff between distance and bandwidth can be seen.

The Pareto front contains only about one-third of the variants. As its shape is similar to Fig. 3(left), we refrain from showing another figure. Among the DYST-Ext variants in the Pareto front, both SHA-3 and ad hoc checksums show quite often. CRC8 shows only seldomly. CRC8 and BCH are often doubles, i.e., they have the same distance and bandwidth as DYST-Basic or -Ext with SHA-3. Quite some variants from the theoretical analysis are not on the Pareto front, indicating that sometimes the simulations gain a little in practice.

D. Taxonomy

Existing publications on network covert channels exclusively focus on network traffic that is live traffic to be modified or generated. Some covert channels also replay traffic recordings enhanced with secret data. *History* covert channels *point* to secret data in a carrier that was transmitted in the past (Fig. 4), i.e., the carrier traffic of the data signal is not

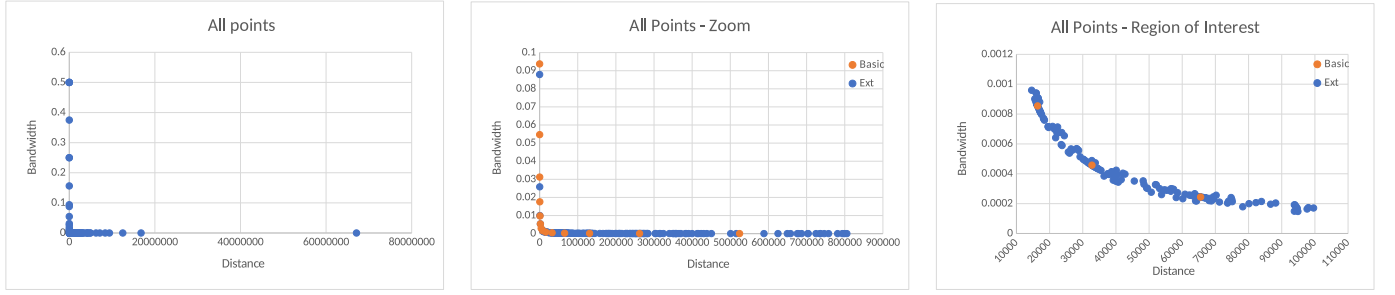


Fig. 3. Simulation results for DYST-Ext (left), and zooms for DYST-Basic and DYST-Ext, respectively. Axes give average distance (in observed packets) and bandwidth (in bit per observed packet), respectively.

altered. Similarly, it would be imaginable to create *prediction* covert channels, which point to anticipated future data. For instance, ARP requests in LANs and sensor value readings in CPS occur on a regular basis and are thus possible to predict. The only difference between prediction and history covert channels is whether they point to old or upcoming data. However, predictions of future traffic are less reliable than pointing to already-seen traffic. For this reason, we solely focus on history channels.

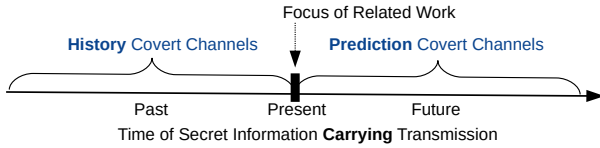


Fig. 4. History and prediction covert channels, differentiated by the secret data carrying transmission they point to.

Definition 2. A **prediction covert channel** operates like a history covert channel but points to *anticipated* data instead of already seen data. ■

The literature differentiates covert channels into active and passive ones (Fig. 5). An active sender generates the traffic in which the secret data is embedded while a passive sender modifies third-party traffic for this purpose. Usually, a passive sender is an intermediate network node, such as a router. The receiving process can also be performed in an active or passive manner. Here, the terminology considers a receiver as active if it is also the destination of the overt traffic. If it passively observes the traffic (which is directed to another hop), the receiver is considered passive.

		Covert Sender		
		Active (generates own overt traffic in which it embeds covert data)	Passive (embeds covert data in overt traffic of third-party nodes)	Fully-passive (utilizes third-party traffic without modifying it)
Covert Receiver	Active (is the destination of the overt traffic)	Active Covert Channel	Semi-passive Covert Channel	Fully-and-semi-passive Covert Channel
	Passive (is not the direct destination of the overt traffic, e.g., a router)	Semi-active Covert Channel DYST's Signal Channel	Passive Covert Channel	Fully-passive Covert Channel DYST's Data Channel

Fig. 5. Categorization of DYST's data and signal channels.

Lamshöft and Dittmann recently added a further differentiation in [31], which is also shown in Fig. 5. They consider covert channels as semi-active if the covert sender is active but the covert receiver is passive. In contrast, a channel is called semi-passive if the covert sender is passive but the covert receiver active.

As the current differentiation between active and passive covert channels does only represent the *signaling* channel of DYST, we add a new category of covert communication, which we call *fully-passive* because of its truly passive handling of third-party traffic (which is not modified). Because of the broadcast nature of the utilized messages, the receiver of DYST's *data* channel is a passive one.

As our channel's sending and receiving processes are decoupled in a way that the sender does not directly address the receiver, our data channel can further be considered as an *indirect* one, while the signaling channel can be considered as a *direct* covert channel.

Finally, a covert channel could also be a *fully-and-semi-passive* one, which is at least theoretically feasible and reflects a channel where the covert receiver waits for pre-defined packets directed to it by some third-party node (fully-passive covert sender). Such a channel could be configured by having DYST operate with directed messages instead of broadcast messages and would be less functional.

IV. EVALUATION

After presenting our implementation and the experimental setups for different scenarios in which we evaluated DYST, we analyze the robustness and detectability of our covert channel. Finally, we formulate and evaluate DYST's application and optimization.

A. PoC Implementation

The PoC for DYST was implemented with Python 3 and utilizes the *scapy* library for eavesdropping legitimate traffic and crafting signal packets. Our DYST implementation utilizes the following packets as they can be received by CS and CR when residing in the same network:

- 1) IPv6 anycast packets with IPv6 destination `ff0*::` or to IPv6 link layer address `33:33:*`
- 2) IPv4 broadcasts to the subnet broadcast-address
- 3) ARP requests to broadcast address `ff:ff:ff:ff:ff:ff`

For hashing, we utilized the SHA3 hash algorithm with a bit length of 512, provided by the Python 3 library *hashlib*. The input values contained the source IPv4 and IPv6 address, depending on the packet type. As the same input of a hash function results in the same hash, we additionally utilized the CS and CR packet receiving timestamp in seconds, resulting in a new hash for the same source addresses each second. As packets are not received at the same time by all devices in a network, we filtered packets that were received at a second fraction lower than 0.05 and higher than 0.95 seconds.

For signaling, we utilized an ARP broadcast request, sent by the CS requesting the MAC address containing the target IPv4 address of an uninvolved third-party system. The CR interpreted this request as the signal to extract the latest hashed value.

B. Scenarios and Testbed

To evaluate DYST under different circumstances, we came up with several scenarios which provide different traffic characteristics. These scenarios are described in detail in the following paragraphs.

a) Scenario 1: University Network: Traffic for this scenario was recorded in a university office network from regular office workstations. We did not use any port mirroring or a prominent location in the network to see how a regular device would see traffic. The environment itself is composed of 75 to 100 devices, around 50 of which are used on a daily basis. The network mostly consists of office laptops, printers, and some smart devices. All major operating systems (Linux, macOS, Windows) are present. For our intents and purposes, the university network resembles that of a company, thus this scenario applies to use cases in both settings.

Example use case: In an APT, an attacker might infect multiple clients in the network and use DYST as a means of internal communication between the compromised clients. If one infected machine got access to an account with higher privileges, it could share the credentials with all other instances in the network for a faster spread of malware. Similarly, DYST could be used as a command and control channel between multiple compromised clients.

b) Scenario 2: Home Network: This scenario represents a typical home network with mixed devices, permanently connected to a WiFi router. The utilized router was a Speedport Smart 3 with current firmware, extended by two mesh repeaters to cover a larger area. In total, up to 30 devices were connected simultaneously, consisting of classical IT devices (three laptops, two raspberries, a network printer, smartphones), IoT devices (SmartTV, vacuum cleaning robot, coffee machine) as well as home automation (various Google Nest Mini). All devices were commonly used and, except for laptops and smartphones, connected permanently to the home network. All devices were connected within one /24 IPv4 subnet.

Example use case: Several compromised smart home devices exchange information under the radar, e.g., to collaboratively collect surveillance data and profile inhabitants. It is not probable that such a network is monitored for covert

channel detection, so a less sophisticated approach will also be applicable. Anyhow, we decided to analyze this scenario because it shows the flexibility of DYST, even if there are few changing devices. In this scenario, especially the throughput of DYST can be optimized as there are no wardens.

Note that further scenarios are imaginable, e.g., journalists exchanging secret information through a WiFi hotspot in a bullet train or at a public airport.

C. Match Distribution

The utilized input parameters of the hash function generate different hashes h_i for each modified bit, i.e., new packet, if the hash function is collision-resistant. These generated hashes h_i are compared by the CS to a specific pattern M (the data it wants to signal), which is constant until a suitable match is found. As explained in the derivation of Eq. (5), the number of matching bits follows a binomial distribution with h trials and a success probability 0.5.

We compare the actual frequency for the number of matching bits in relation to the total number of hashes generated to the expected results in Fig. 6 for both scenarios. Therefore, we searched for one specific pattern equal to 8 bits and expected the distribution to follow the binominal distribution with $N = 8$ and $p = 0.5$, which is represented by the grey histogram. Further, we assume that the more hashes are observed, the more the actual distribution will follow the expected distribution. The black and red bars represent the actual distribution of matching bits in the home and university network scenarios, respectively. Both actual distributions follow the expected distribution, however, the university network scenario differs slightly. This can be explained by the number of observed hashes (89,959 observed hashes for the home network; 2,075 for the university network), drawing both assumptions correct. This points out our hash-generation methodology is correct.

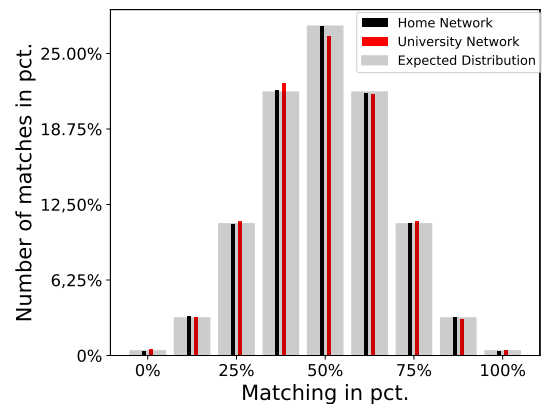


Fig. 6. Match Distribution

Besides the evaluation in our testbeds, also the number of matches for different bit lengths is evaluated. As the CS is searching for a 100 percent match in DYST-Basic, we calculated $P_h(X \geq h)$. As for DYST-Ext also partial matches can be utilized, we add $P_h(X \geq 0.8 \cdot h)$ for at least 80% matching bits, both calculated by Eq. (5).

The results of example experiments for the home network scenario are presented in Tab. I. The 8-bit and 16-bit experiments were performed in a live scenario, while the 12-bit and 21-bit match distributions were simulated. The results indicate that for $h = 8$ bits, slightly more hashes than expected show h matching bits, while for at least 80% matching bits, slightly less were observed. According to the deviation, 281 perfect matches should be detected, while actually 343 were detected. For the DYST-Ext mode, 3,232 potential matching hashes should be detected, while actually 3,162 were detected. For our $h = 12$ bit experiment, the hit rate for both 100 and (at least) 80 percent matches slightly performed worse than expected, resulting in 150 matches instead of 164 and 1,068 instead of 1,086, respectively. For $h = 16$ bit and a 100 percent match, the observed rate neared the expected value, while the number of actual 80 percent matches was slightly higher than expected. There should have been 5 matches, while actually 5 had been found for DYST-Basic and for DYST-Ext potential 2,379 hashes with at least 80 percent match should be observed, while 2,413 were actually detected. For our $h = 21$ bit experiment, no matches were found. As 0.43 packets are expected for the number of observed packets, the expectation is met. Further, 152 packets had at least 80% matching bits, while there should have been 170 packets.

TABLE I
EXPECTED AND ACTUAL MATCH RESULTS (HOME NETWORK)

h	8 bits	12 bits	16 bits	21 bits
# observed packets	89,959	56,105	226,488	43,325
$P_h(X \geq h)$	0.3125%	0.2930%	0.0024%	0.0001%
freq. pkts w. 100% match	0.3813%	0.2673%	0.0022%	0.0000%
$P_h(X \geq 0.8 \cdot h)$	3.5937%	1.9360%	1.0504%	0.3917%
freq. pkts w. $\geq 80\%$ match	3.5160%	1.9035%	1.0654%	0.3508%

D. Robustness

The main concern for the robustness of DYST lies in which messages are seen and interpreted for signaling. As DYST only uses legitimate network packets, it relies on the general robustness of network transmissions (e.g., Ethernet frame checksums) and timestamps (ensured by time synchronization mechanisms like NTP for example). If the same packet arrives at CS and CR, we can assume that the content will be the same, resulting in the same input parameters for the hashing function. We do have other concerns about the robustness of DYST that need to be addressed:

- CS and CR must receive the same *packets of interest* (PoI).
- CS and CR must receive the PoIs in the same order.
- Consecutive PoIs must have a sufficient delay to allow reliable signaling.

a: To address this issue, we have to carefully select which packets are used by DYST to ensure that both CS and CR receive the same packets. Depending on the deployment scenario, we might choose different sets of packets to achieve this goal: In a local scenario like a university network or an open Wi-Fi like a café, we can focus on local broadcast packets. If DYST is used between two routers, we can use

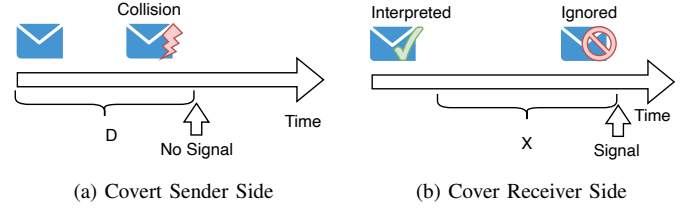


Fig. 7. Robustness Measures

our knowledge about the routing topology to filter for packets that will pass through both CS and CR. Similarly, if both CS and CR are part of the same multicast group, one could filter for packets from that group to ensure synchronization. Moreover, there is a tolerance regarding the received PoI. As only a fraction of the packets are actually used for signaling, a packet received by CR but not received by CS would simply not be checked for a match and thus also not used as a carrier. However, if CS receives a packet and both of the following conditions are met: CR does not receive the packet and CS actually points to this packet for signaling, then CR might receive an incorrect message as it interprets the wrong packet (see point b)). All in all, it is possible to choose a robust set of packets to be used with DYST, with only a little prior knowledge about the deployment scenario.

b: This variable is outside the control of DYST as we cannot influence the routing or buffering behavior of other parts of the network. We performed an evaluation in a home network by running DYST between two different clients on the same network. During our evaluation, we observed significant problems when testing our scripts. In our first test, 3 of 65 characters of the message were transmitted correctly. This was due to the fact that a significant portion of the PoIs did not arrive in the same order for the CS and the CR. These PoI packets were in the wrong order because of their tight succeeding timing and different networking delays for CS and CR. Therefore, two additional configuration parameters for DYST control the mandatory delay between received packets to reduce this issue: If the CS receives two or more PoIs in less than D milliseconds, the CS will ignore all PoIs received in that timeframe. So only isolated PoIs will be considered for DYST. If only one PoI is received in D milliseconds, the CS will send out the signal. The CR will ignore all PoIs that arrived less than X milliseconds before the signal and only interpret earlier PoIs. This gives the CS enough time to calculate and send the signal without risking a race condition (see part c) below). Fig. 7 illustrates this process. Fig. 7a, shows the side of the covert sender. This example uses two packets that arrive close to each other, which leads to them being ignored by the CS and not considered for DYST. Fig. 7b shows the receiver-side. Here we can see that the receiver ignored a packet that arrived too close to the signal and instead interpreted the older one. This will decrease the potential throughput, as we ignore more packets that could potentially be useful. But we significantly decrease potential errors in cases where packets arrive in a different order at the CS as at the CR. Varying scenarios will require differently

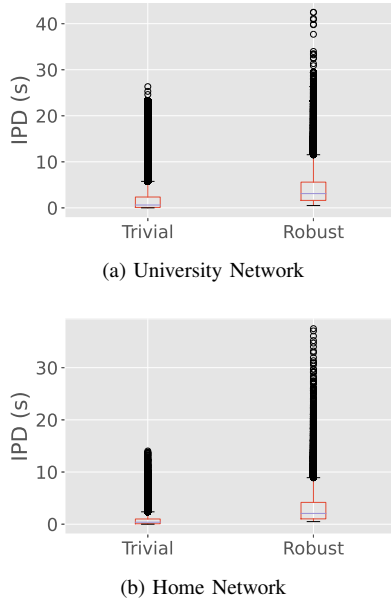


Fig. 8. Comparison of inter-packet delays (IPDs) between PoIs (SIMULATED)

tuned values for D and X , see Sect. IV-F. Additionally, we can counter possible errors with an error-correcting code that covers multiple transmissions. Different approaches are possible, simply transmitting the same message multiple times and taking a majority vote on gained hash values, can increase robustness.

c: Similarly to b), DYST might encounter errors if many PoIs arrive in a short amount of time. The CS might receive a PoI that creates a hit. While the CS is evaluating and preparing to send out the signal, another PoI arrives at the CR even before the signal from the CS reached the CR. In such a case, the CR would interpret the latest PoI and not the correct one. This error source is also countered by the option X of the multistage delay introduced in b). Since DYST only considers PoIs that are isolated and the CS has X milliseconds to perform the signaling, it is far less likely for this race condition to appear for any PoIs that can be seen by CS and CR. Similar to b), a larger delay will result in a lower bandwidth but higher robustness. Depending on the scenario, drastically different configurations for D and X are possible. If CS and CR are both routers with a fixed route between them, it is easy to see that the CS will have better knowledge about the order and delays in which the CR will receive the PoIs and can therefore choose a lower value for D and X .

Impact of Robustness: We evaluated the timing between PoIs in both scenarios to get an overview of network behavior and the impact of our robustness measure. Fig. 8 shows the plots for the different scenarios.

We can see that both scenarios show a generally similar picture: the mean value for the inter-packet delays (IPDs) of the PoIs for DYST-Basic is close to 0 with a significant number of outliers. The outliers are beneficial for the robustness of DYST, as they separate PoIs. If we look at the delays with the robustness filter ($D = 0.5s$, the simulation only happened for the CS side), we again see a similar image for all scenarios. We

TABLE II
ROBUSTNESS EVALUATION - MATCHING TRANSMITTED CHARACTERS

	Non-robust Matches	Robust Matches
University Network	100%	100%
Home Network	4%	56%

TABLE III
ROBUSTNESS EVALUATION - TOTAL POIs (SIMULATED)

	Non-robust	Robust ($D = 0.5s$)	Fraction (%)
University Network	8,130	4,595	56.5%
Home Network	79,319	12,689	15.9%

generally see higher delays between PoIs, as DYST ignores some PoIs and therefore the delays between evaluated PoIs are higher. This means that DYST will have fewer packets at its disposal to transmit the message, but we gain in reliability (see Sect. IV-F).

We evaluated the effectiveness of our robustness approach in the home network and university network scenarios. Tab. II shows the results for two scenarios (home and university network). For each scenario, we conducted a test run with and without robustness measures and recorded the percentage of characters that were correctly transmitted.

We can see that the home network had significant problems without robustness measures, while the university network setup had no problems during our tests. The university network setup showed very little activity during our robustness tests, which resulted in a reliable transmission but also in a low bandwidth (8 characters in 8 hours). The home network setup had drastically more activity and higher bandwidth (48 characters in 7 hours), which in turn resulted in a significantly less reliable transmission. Initially, we tested with $D = 0.3s$ and $X = 0.015s$ which resulted in ca. 31% correctly transmitted characters. We further tested the parameters $D = 0.5s$ and $X = 0.3s$, which resulted in 56% correctly signaled characters. We can therefore see that the home network setup benefits from more aggressive configurations. However, the robustness would increase using a higher D or additional robustness measures, such as the redundant transfer of secret messages.

In Tab. III, we show how many PoIs were observed in a simulated offline run. We used real recordings and ran the pcaps through an offline version of DYST (only CS side is simulated). We can see that in all scenarios, the number of usable PoIs is significantly reduced. This means there will be fewer PoIs available for DYST, which in turn can reduce the potential bandwidth of the covert channel. It is noticeable that the home network setup suffered more than the university network setup, this can be explained by the higher activity in the home network compared to the university network. This again explains the worse performance in the home network without robustness measures.

On the other hand, a reduction of possible bandwidth aids the undetectability, by spreading signals even further apart.

E. Detectability

To evaluate the detectability of DYST, we gathered legitimate reference and covert channel recordings from two different scenarios and with six different configurations for the covert channel (1 and 2 byte basic mode, 1 and 2 byte robust mode, and 1 and 2 byte extended mode). We performed recordings in two different networks, a home network and a university network (see Tab. IV for an overview).

TABLE IV
OVERALL RECORDED TRAFFIC

		# Pkts	# ARP Requests	Rec. Time
Legitimate	Home Network	17,608,213	8,619,952	40 Days
	University Network	2,096,387	218,707	15 Days
Covert	Home Network 1	5,604,205	3,023,422	11 Days
	University Network 1	5,380,286	522,451	56 Days

As the data channel itself cannot be detected, a defender relies on detecting the signal channel. Since our DYST implementation uses ARP requests for signaling, our detection focuses on the IPDs of ARP requests.

1) *KS-Test*: To gauge the potential detectability, we chose the KS-Test [32] as a general measure of similarity between the ARP IPD distribution of two recordings. In addition to our original recordings, we also filtered the covert channel recordings to remove any signals produced by the covert channel to produce a second (but synthetic) source of legitimate traffic. We then performed a cross validation between all possible recordings of a scenario (home and university networks). This provides us with 3 different classes of combinations, which we considered important for our analysis: (1) covert vs. filtered recordings, (2) pairs of legitimate recordings, and (3) covert vs. legitimate recordings. To get a better understanding of the detectability, we focused on several combinations, which we describe separately.

First, we compared different *legitimate* recordings against each other, while excluding exact matches. For this, we used several legitimate recordings from the home network. This gives us a mean p-value of $3.84e^{-11}$ with a standard deviation of $4.01e^{-10}$ and a mean D-value of 0.19 with a standard deviation of 0.13. The results for the legitimate university network scenario are almost exactly the same. This points towards a significant difference between all the legitimate recordings. With that, we can already see that legitimate traffic drastically varies depending on the time of day and the activity of participating nodes that do not follow repetitive behavior. This already points to a low possibility of detection.

Next, we compared the *covert channel* recordings with the corresponding *filtered covert channel* recordings. This gives us a mean p-value of 0.99 with a standard deviation of 0.04 and a mean D-value of 0.0008 with a standard deviation of 0.001. This on the other hand shows that our covert channel barely alters the characteristics compared to legitimate traffic.

If we compare *covert channel* recordings with *legitimate* recordings, we obtain a mean p-value of $1.53e^{-95} \sim 0$ with a standard deviation of $1.12e^{-94} \sim 0$ and a mean D-value of $1.87e^{-01}$ with a standard deviation of $1.31e^{-01}$. These results again point towards significant differences between the

two scenarios. However, these differences are comparable to those of two legitimate recordings.

Fig. 9a shows the p-values of the KS-tests and we can see a strong similarity of the CC vs. filter scenario and a strong dissimilarity for the other two scenarios. Fig. 9b, 9c and 9d show exemplary histogram plots for the ARP request IPDs for each of the considered scenarios from the home network setup. Again, we can determine a high level of similarity between the covert channel recording and the filtered recording and slight differences between two legitimate recordings as well as between covert and legitimate recordings.

These three results combined point towards an almost impossible detection of DYST, as the differences between different legitimate recordings are larger than the difference between covert channel and filtered recordings.

2) *Compressibility Score*: In addition to the KS-test we also used a widely known detection method from covert channel research: the compressibility score as proposed by Cabuk et al. [16]. Again, we used the IPDs of ARP requests for the detection and focused on the same three classes of combinations with the same recordings. To calculate the compressibility score, we divided the recordings in windows of fixed length (1,000 IPDs). Each window was then transformed into a string representation, which was compressed using `gzip`. The final compressibility score κ for a window is the compression ratio between the original string and the compressed string.

Fig. 10a shows the compressibility scores for all recordings. Here we can see a nearly identical values for legitimate and covert channel recordings, which already points towards a challenging detectability of DYST. Figs. 10b, 10c and 10d show exemplary histograms with pairwise plots for the distributions of the compressibility scores of the home network. As can be seen, we found a large overlap of compressibility scores when comparing covert channel and filtered recordings. This again points towards a strong similarity of the two recordings. The comparison of two legitimate recordings of the home network shows significant differences in the distributions while we would expect two matching plots, indicating a high dependence on the current status of a network (daytime etc.). Moreover, the comparison between the covert channel and the legitimate recording shows even smaller differences than the two legitimate recordings. When determining optimal thresholds for the detection of DYST, our experiments revealed that the compressibility metric (and the related AUC scores for ROC charts) fluctuated based on time of day, network load or other factors, rather than the presence of DYST. This is rooted in the fact, that DYST sends only very few messages. Thus, similar to the KS-test, we observed that there is no clear threshold for the κ -value to discriminate between legitimate and covert channel recordings. This fact leads to the conclusion that DYST is not detectable with the compressibility score.

3) *Evaluation of Different Covert Channel Configurations*: We also evaluated the detectability of DYST when only focusing on a single configuration at a time (e.g., 1 byte basic or 2 byte ext. modes). We found that the compressibility score performed no different than before, still producing overlapping κ -values. Similarly, the KS-test performed comparable.

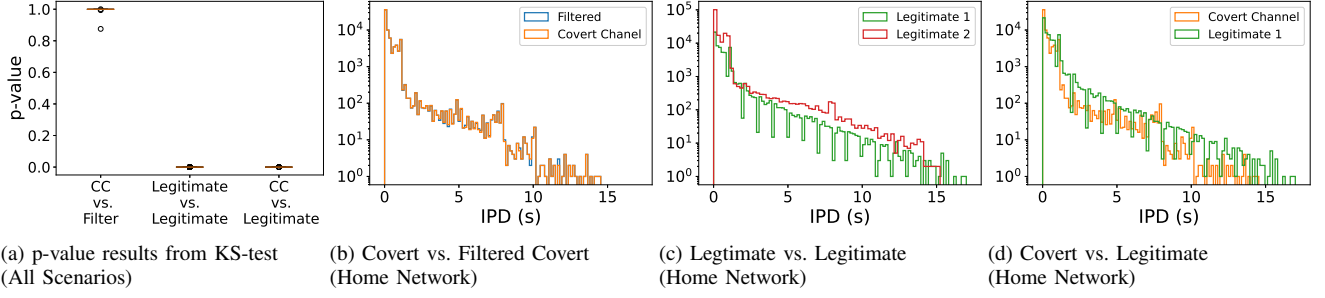


Fig. 9. Overview of legitimate and DYST traffic's characteristics, with detailed examples for IPD values of the home network.

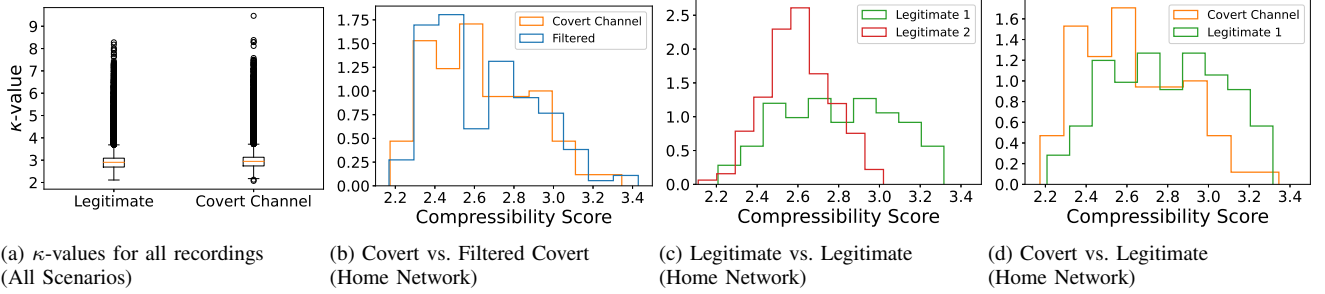


Fig. 10. Overview of legitimate and DYST traffic's compressibility scores, with detailed examples for the home network scenario.

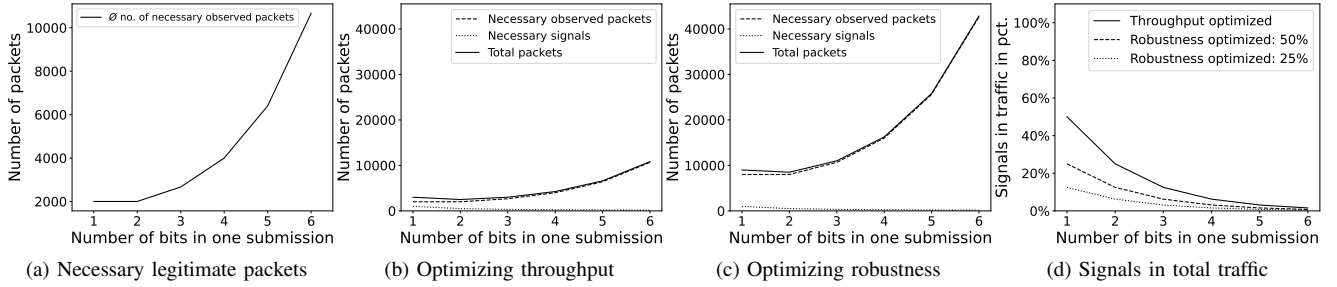


Fig. 11. Transmission of 1,000 bits with DYST

We conclude that DYST is currently undetectable, which is mostly rooted in the small number of signal packets sent.

F. Optimization

Fig. 11 presents the transmission of 1,000 bits with DYST with 1 to 6 bits transferred at once. The theoretical number of hashes necessary to transmit a Message M with DYST-Basic can be calculated by dividing $\frac{\text{len}(M)}{h}$ and $P_h(X \geq h)$, resulting in $\text{len}(M) \cdot 2^h / h$. The number of necessary packets to transmit a Message M with a length of 1,000 bits is visualized in Fig. 11a and shows that the more bits shall be submitted at once, the more packets need to be observed. The number of packets grows exponentially with the number of bits to transmit in one chunk. This represents one packet for each observed match, disregarding robustness and stealthiness considerations. Further, the number of necessary signals decreases with the number of bits transferred in one chunk. This leads to a tradeoff between stealthiness and throughput. The stealthiness also increases with robustness, as for a robust transmission from CS to CR some packets need to be ignored

as already described. Henceforth, the more robust the channel, also the stealthier DYST will be, however, the throughput suffers. This trade-off is also presented in Figs. 11b-11d for the transmission of the message M consisting of 1,000 bits with a maximum number of 6 bits signaled at once. In the case of a throughput-optimized DYST implementation, for each matching hash, a signal is sent. Fig. 11b visualizes this configuration where the number of signals to be sent decreases as the number of bits encoded in one matching hash increases (dotted line), and nears 1 for a theoretical signal, pointing at 1,000 bits at once. The number of necessary packets to observe increases (dashed line) as the probability of a match also decreases because the total number of packets in a recording increases. These are represented by the solid line (showing the sum of necessarily observed packets and necessary signals). For a robust approach, we assume in our example that only each fourth packet fits our previously described requirements (see Fig. 11c). The number of necessary packets to observe increases while the number of signals stays constant, compared to the throughput-optimized scenario. Thus, compared to the

total number of packets observed, *fewer* signals are included in an experiment. The share of signals in percent is shown in Fig. 11d. The solid line represents a throughput-optimized approach, while the dashed line shows a robust approach where each second packet fits the robustness requirements. The dotted line visualizes the approach where each fourth packet fits these requirements and equals the scenario in Sect. IV-D. The share of signals in percent decreases with the number of packets utilized for signaling and with the number of bits signaled at once. This leads to the conclusion that the fewer packets are utilized and the more bits are signaled at once, the less likely a detection. If CS and CR fear the presence of a warden, they can simply ignore each certain number of packets or increase the number of bits signaled at once to decrease the noticeability of a signal.

V. DISCUSSION

The concept behind DYST, observing historic traffic and letting a receiver know at which point in time historic data should be interpreted, can also be applied to other scenarios. For instance, a social media service, where legitimate users submit postings, such as *Facebook*, *Twitter*, or any form of blog generates a massive amount of content. As long as the content is hashed in a similar way, it can be transformed into random bitstrings. The covert sender could signal the occurrence of a match by posting timed replies or comments on these platforms. However, in comparison to network environments, such replies are not ephemeral and might be used for later anomaly detection of CS's behavior. Also, cloud services like *Dropbox* or *Google Drive* may be exploited in similar ways by hashing details of modified files. Signaling for these scenarios is not bound to the same service, which means that signals can be sent out-of-band, for example via a different platform or a network protocol. Similarly to social media and web content, DYST could also be applied for stealthy communication between local processes of a secure operating system by monitoring hardware events.

As evaluated in the previous sections, existing detection methods do not uncover the flow of covert information if DYST is implemented in a network environment. However, under certain conditions a throughput-optimized version of DYST, with fewer bits signaled at once, may be detected as the share of signals in total traffic will be higher and is significant for the detection. This can also be considered a limitation of information exchange as the throughput is constrained. If a non-existent packet destination is chosen, this may cause anomalies, leading to detection. The covert channel can further be limited or even eliminated if CS and CR are forced to switch collision domains regularly, however, this may not be a practical scenario for various network environments. Further, an active defender might send distorting signals. However, this leads to the problem that more potential matching hashes can be created, increasing a potential throughput. An active defender may alternatively collect each exploitable data packet and send them in convoys, though this will cause delays and may influence a network environment massively. All discussed countermeasures need further investigation and need to be evaluated in the future.

Replicability Measures: Reproducing scientific experiments is considered a challenge. For this reason, we ensured that the experiments on DYST are replicable. To this end, we made sure that the code of DYST is available: <https://github.com/NIOsAT/DYST>. The code for simulating different variants of DYST-Basic and DYST-Ext to compute (distance, bandwidth) tuples is also included, together with a spreadsheet that contains all raw simulation results and the Pareto front data, plus "readme" files to explain the handling.

VI. CONCLUSION

We presented DYST, the first covert channel that sends secret messages through the content of unaltered legitimate network traffic. We believe that this channel represents a new class of covert channels that we call *history covert channels*. In comparison to state-of-the-art methods, history covert channels do not rely on the imperfect modification or imitation of legitimate traffic. However, our covert channel still requires sending one signaling packet, but this can be any type of packet, such as a legitimate and plausible ARP request. We presented two feasible encoding strategies and implemented DYST in two network environments. Our results for this initial work indicate a limited but variable throughput dependent on the number of bits signaled at once. Our implementations' robustness increases with confining potential race conditions for data packets. We also analyzed the detectability of DYST and have shown that traditional heuristics used in network covert channel research are unable to provide satisfying results. Further, we have shown that the stealthiness of DYST increases with the robustness, at the expense of throughput. We also presented feasible application scenarios besides network protocol-based approaches and discussed potential countermeasures. In future work, we plan to analyze further ways to implement history covert channels, e.g., through social media, increasing the sending performance and develop additional countermeasures.

REFERENCES

- [1] B. W. Lampson, "A note on the confinement problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.
- [2] Department of Defense (DoD), "Trusted computer system evaluation criteria," 1985, DOD 5200.28-STD.
- [3] S. Zander, A. Grenville, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys and Tutorials*, vol. 9, pp. 44–57, 2007.
- [4] B. Carrara and C. Adams, "Out-of-band covert channels—a survey," *ACM Comput. Surv.*, vol. 49, no. 2, jun 2016. [Online]. Available: <https://doi.org/10.1145/2938370>
- [5] D. Barradas and N. Santos, "Towards a scalable censorship-resistant overlay network based on WebRTC covert channels," in *Proc. 1st Int. Workshop on Distr. Infrastr. for Common Good*. ACM, 2020, pp. 37–42.
- [6] A. Mileva and B. Panajotov, "Covert channels in TCP/IP protocol stack-extended version," *Open Computer Science*, vol. 4, no. 2, pp. 45–66, 2014.
- [7] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-based survey and categorization of network covert channel techniques," *Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 50:1–50:26, 2015.
- [8] S. Wendzel, W. Mazurczyk, and G. Haas, "Don't you touch my nuts: Information hiding in cyber physical systems," in *IEEE Security and Privacy Workshops (SPW'17)*. New York, NY: IEEE, 2017, pp. 29–34.
- [9] P. Krishnamurthy, F. Khorrami, R. Karri, D. Paul-Pena, and H. Salehghaffari, "Process-aware covert channels using physical instrumentation in cyber-physical systems," *IEEE Transactions on Information Forensics and Security*, vol. 13 (11), pp. 2761–2771, 2018.

- [10] M. Hildebrandt, R. Altschaffel, K. Lamshöft, M. Lange, M. Szemkus, T. Neubert, C. Vielhauer, Y. Ding, and J. Dittmann, "Threat analysis of steganographic and covert communication in nuclear I&C systems," in *International Conference on Nuclear Security: Sustaining and Strengthening Efforts*. Vienna: IAEA, 2020.
- [11] K. Lamshöft, T. Neubert, C. Krätzer, C. Vielhauer, and J. Dittmann, "Information hiding in cyber physical systems: Challenges for embedding, retrieval and detection using sensor data of the SWAT dataset," in *Proc. ACM Workshop on Information Hiding and Multimedia Security*, ser. IH&MMSec '21, 2021, p. 113–124.
- [12] J. Millen, "20 years of covert channel modeling and analysis," in *Proc. 1999 IEEE Symposium on Security and Privacy*. New York, NY: IEEE, 1999, pp. 113–114.
- [13] Q. Xu, H. Naghibijouybari, S. Wang, N. Abu-Ghazaleh, and M. Annam, "GPUGuard: Mitigating contention based side and covert channel attacks on GPUs," in *Proc. Int. Conf. Supercomputing*. New York, NY, USA: ACM, 2019, p. 497–509. [Online]. Available: <https://doi.org/10.1145/3330345.3330389>
- [14] M. Urbanski, W. Mazurczyk, J.-F. Lalande, and L. Caviglione, "Detecting local covert channels using process activity correlation on Android smartphones," *International Journal of Computer Systems Science and Engineering*, vol. 32, no. 2, pp. 71–80, 2017.
- [15] K. Block, S. Narain, and G. Noubir, "An autonomic and permissionless Android covert channel," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '17. New York, NY, USA: ACM, 2017, p. 184–194. [Online]. Available: <https://doi.org/10.1145/3098243.3098250>
- [16] S. Cabuk, "Network covert channels: Design, analysis, detection, and elimination," Ph.D. dissertation, Purdue University, 2006.
- [17] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. USA: USENIX Association, 2006.
- [18] R. J. Walls, K. Kothari, and M. Wright, "Liquid: A detection-resistant covert timing channel based on IPD shaping," *Computer Networks*, vol. 55, no. 6, pp. 1217–1228, 2011. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128610003580>
- [19] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-based covert timing channels: Automated modeling and evasion," in *Recent Advances in Intrusion Detection*. Springer, 2008, pp. 211–230.
- [20] F. V. Yarochkin, S.-Y. Dai, C.-H. Lin, Y. Huang, and S.-Y. Kuo, "Towards adaptive covert communication system," in *Proc. 14th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2008)*. Los Alamitos, CA: IEEE Computer Society, 2008, pp. 153–159.
- [21] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2009.
- [22] S. Wendzel and J. Keller, "Hidden and under control," *Annals of Telecommunications*, vol. 69, no. 7, pp. 417–430, 2014.
- [23] L. Caviglione, W. Mazurczyk, and S. Wendzel, "Advanced information hiding techniques for modern botnets," in *Botnets: Architectures, countermeasures, and challenges*. Taylor & Francis, 2019, pp. 165–188.
- [24] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert channel detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 12, no. 4, pp. 22:1–22:29, April 2009.
- [25] —, "IP covert timing channels: design and detection," in *Proceedings of the 11th ACM conference on Computer and communications security*, ser. CCS '04. ACM, 2004, pp. 178–187.
- [26] V. Berk, A. Giani, and G. Cybenko, "Detection of covert channel encoding in network packet delays," Department of Computer Science - Dartmouth College, Tech. Rep., 2005.
- [27] X. Zhang, Y.-A. Tan, C. Liang, Y. Li, and J. Li, "A covert channel over volte via adjusting silence periods," *IEEE Access*, vol. 6, pp. 9292–9302, 2018.
- [28] A. Ovidia, R. Ogen, Y. Mallah, N. Gilboa, and Y. Oren, "Cross-router covert channels," in *13th USENIX Workshop on Offensive Technologies (WOOT 19)*. Berkeley, CA: USENIX Association, Aug. 2019. [Online]. Available: <https://www.usenix.org/conference/woot19/presentation/ovadia>
- [29] G. C. Clarke, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communication*. New York: Springer Science+Business Media, 1981.
- [30] L. R. Vermani, *Elements of Algebraic Coding Theory*. London: Chapman & Hall, 1996.
- [31] K. Lamshöft and J. Dittmann, "Assessment of hidden channel attacks: Targetting Modbus/TCP," *IFAC-PapersOnLine*, vol. 53, no. 2, 2020.
- [32] F. J. Massey, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Stat. Assoc.*, vol. 46, no. 253, pp. 68–78, 1951.

VII. BIOGRAPHIES



Steffen Wendzel is a professor at Hochschule Worms, Germany, where he is also the scientific director of the Center for Technology and Transfer (ZTT). In addition, he is a lecturer at the Faculty of Mathematics & Computer Science at the Fern-Universität in Hagen, Germany, from which he also received his PhD (2013) and Habilitation (2020). Before joining Hochschule Worms, he led a smart building security research team at Fraunhofer FKIE in Bonn, Germany. Steffen (co-)authored more than 170 publications and (co-)organized several conferences and workshops and special issues for major journals, such as Trans. Indust. Inf., Secur. & Priv. and Future Gen. Comp. Syst. His major research focus is on covert channels, network steganography, scientific taxonomy, and IoT security. Website: <https://www.wendzel.de>.



Tobias Schmidbauer is a PhD student at Fernuniversität in Hagen, Germany. Before, he studied computer science in public management (Verwaltungsinformatik) at Hochschule Hof / Hochschule für den öffentlichen Dienst Hof, Germany and received his diploma in 2016. Thereafter, he started studying practical computer science in part-time at Fernuniversität in Hagen, Germany, and graduated with an MSc degree in 2019. Alongside his studies, he worked from 2007 until 2017 for the datacenter of the tax administration of the German federal state of Bavaria. Since 2017, he works full-time for the Bavarian State Office for Information Security as a penetration tester. In addition, he was deputy information security officer of the Bavarian State Office for Information Security until the end of 2020 and has held this position in full-time since the beginning of 2021.



Sebastian Zillien is a researcher and PhD student at the Hochschule Worms, Germany, where he works for the project SIVERT (Secure & Intelligent Visualization- & Real-time Reconstruction Methods (for pCT)). He received his BSc in Applied Informatics in 2018 and his MSc in Mobile Computing in 2020 from Hochschule Worms. His research focus includes network and protocol-level security, anomaly detection, covert channels and reliability. He has worked on multiple projects in his field and has had publications, among others, at IFIP SEC, NordSec and in the Journal of Universal Computer Science (JUCS).



Jörg Keller Jörg Keller is a professor of computer engineering at FernUniversität in Hagen, Germany, where he leads the Parallelism & VLSI research group since 1996. He received MSc and PhD degrees and habilitation from Saarland University, Saarbrücken, in 1989, 1992, and 1996, respectively. His research interests include network steganography, cryptographic primitives for embedded systems, energy-efficient and fault-tolerant parallel computation, and blended and virtual laboratories. He is author or co-author of 2 books and more than 180 refereed articles in journals and conference proceedings. He has co-organized numerous conferences and workshops, and special issues for Journal of Universal Computer Science. He is a member of the editorial boards of Journal of Cybersecurity & Mobility and Journal of Universal Computer Science. His research group website: <https://feu.de/pv/en>