Strengthening Anonymous Communication via Passive Participation

Abstract: Anonymous communication networks (ACNs) are basic building blocks for obtaining or exchanging data in a privacy-preserving manner. ACNs suffer from a bootstrapping problem: having few users leads to a small anonymity set, which renders the ACN unattractive. We propose a system, CoverUp, that tackles the bootstrapping problem for ACNs. The key idea is to draw in non-ACN users from a collaborating website to connect to an ACN after an informed consent via a JavaScript snippet, thereby triggering them to passively participate (as passive participants). CoverUp implements a privacy-preserving broadcast with a downlink rate of 10 to 50 Kbit/s that renders the traffic of these passive participants indistinguishable from active participants. If this broadcast is accessed via an ACN with a constant-rate traffic pattern, CoverUp contributes active participants with legitimate-looking traffic, thus helping in bootstrapping the ACN. To protect active participants from potentially incriminating broadcast-data, an additional application is needed to extract any information from CoverUp's broadcast. The indistinguishability guarantee of CoverUp for broadcasts holds against global network-level attackers that control everything except for the user's machine. In addition, as long as active participants do not change their surfing behavior on these websites due to CoverUp, they hide their participation time, i.e., do not leak the time at which they listen to the broadcast, which counters intersection and statistical disclosure attacks. As passive participation raises ethical and legal concerns for the collaborating websites and the participants, we discuss these concerns and describe how they can be addressed.

We extend CoverUp to bi-directional point-to-point communication (e.g., messengers) with an up- and downlink rate of 10 to 50 Kbit/s. Bi-directional CoverUp can offer users of ACNs with constant-rate traffic an additional entry point that hides their participation time (as above). The indistinguishability for bi-directional CoverUp requires the integrity of the JavaScript snippet, for which we introduce a trusted party. We give evidence that with a latency of 3 seconds (including the random delays) the timing leakage is undetectable, even after a year of continual observation. As long as the timing leakage is undetected, bi-directional CoverUp achieves the same properties as for broadcasts against network-level attackers that control everything except for the user's machine and the trusted party.

Keywords: keywords, keywords

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

Anonymous communication networks (ACNs) are essential building blocks for obtaining or exchanging data in a privacy-preserving manner. Existing ACNs suffer from two major weaknesses. First, traffic-correlation-resistant ACNs [29, 43, 53, 54, 67, 69] have a bootstrapping problem: few users mean a small anonymity set, and the small anonymity set renders the ACN unattractive, which in turn leads to few users.

Second, according to the revelations, surveillance is mostly based on meta-data, such as source IP, destination IP, timestamps, and size of the data. While anonymous communication networks (ACN) are designed to hide such metadata, even the strongest ACNs from the literature [41, 61, 67, 69] do still reveal against global network-level attackers the *participation time*: when and the fact that a user connects to an ACN, in particular, the intention to use the ACN. Participation in an ACN can appear suspicious and this participation time can be used in long-term statistical disclosure attacks to re-identify the user, which degrades the anonymity properties of an ACN to pseudonymity [45, 47].

Both these weaknesses can be addressed by drawing in passive users to participate in an ACN (as *Passive Participants*), while ensuring that their traffic is indistinguishable from active ACN participants. Legitimate-looking traffic from fresh and legitimate IPs to the ACN helps in bootstrapping an ACN by increasing the anonymity set of the ACN. In addition, as long as active participants do not change their surfing behavior on these websites, they hide their intention to use the ACN and can deny their participation, which would, e.g., counter intersection attacks.

We introduce CoverUp, which implements a privacy-preserving broadcast (or message feed) with a downlink rate of 10 to 50 Kbit/s that renders the traffic of these passive participants indistinguishable from active ones. Such message feeds are suited for the transmission of information that a user does not want to be caught reading (e.g., sensitive medical information, leaked documents, or a leaked e-mail list of an incriminating web service). CoverUp triggers visitors of collaborating highly accessed websites after an informed consent to connect to a broadcast service by using a JavaScript-snippet (JSS), thus triggering the user into passively requesting the broadcast. We protect passive participants from incriminating data by enforcing that a participant's machine never contains enough data chunks to retrieve any information. Active listeners to the broadcast additionally install an application that collects these data chunks and extracts the broadcast. CoverUp renders active and passive participants indistinguishable against global network-level attackers that control everything except for the participant's machine. If this broadcast is accessed via an ACN with a constant-rate traffic pattern, CoverUp contributes passive participants with legitimate-looking traffic, thus increasing the ACNs anonymity set and helping in bootstrapping the ACN. In addition, as long as the active participants do not change their surfing behavior on these websites due to CoverUp, they can deny their participation in the broadcast, i.e., the time at which they listen to the broadcast, which, e.g., counters intersection attacks.

We extend CoverUp to bi-directional point-to-point communication (e.g., messengers), for ACNs with strong anonymity and constant-rate traffic. Additionally, passive participants send dummy traffic to the ACN, and active users install a browser extension that replaces dummy traffic with upstream data. To counter the browser extension's timing leakage, we add random delays to the JSS. As a malicious JSS can be used to detect the browser extension, we introduce an additional trusted party that serves the JSS. We give evidence that with a latency of 3 seconds (including the random delays) the timing leakage is undetectable, even after half a year of continual observation. As long as the timing leakage is undetected, CoverUp for bi-directional communication achieves the same properties as for broadcasts but against a network-level attacker that controls everything except for the user's machine and the trusted party. As long as the active participants do not change their surfing behavior on these websites due to CoverUp, bi-directional CoverUp offers users of ACNs with constant-rate traffic and strong anonymity an additional entry point with which they can deny any participation in the ACN.

Previous work [42, 63] outlined how to use passive participation to achieve a (theoretically) deniable uplink channel in the browser. However, that work left three major challenges unsolved. (*i*) How to construct a downlink connection (using the browser) that relays data to an external program (CoverUp-Tool) with low latency and with minimal timing leakage? (*ii*) How to relay data from an external program to the uplink connection (using the browser) with low latency and with minimal timing leakage. (*iii*) How much timing leakage does an implementation of this approach entail?

1.1 Summary of contributions

- We design a deniable uni-directional channel through an ACN to a broadcast server, the *CU:Feed*, that can deliver data to an external application (challenge (*i*)). This channel tolerates global network-level attacker that control all the parties except for the users' machines. By involving passive participants, we tackle the bootstrapping problem.
- We extend the feed to realize a deniable point-to-point bi-directional channel, the *CU:Messenger*, that can connect to any external application (CoverUp-Tool) via native messaging (challenge (*ii*)). This channel implements a denial entry point for constant-rate ACNs that provide strong anonymity. The channel has minimal timing leakage and tolerates global network-level attackers and a malicious entry server.
- For both channels, we implement prototypes and carefully minimize the timing leakage (challenge (*iii*)). The prototype includes a dummy entry server, CoverUp server that serves the JSS, CoverUp application that provides a display for the feed and a messaging interface and a browser extension that intercept and replaces noise data and communicate with the CoverUp application. A naive implementation would cause a significant timing leakage. Our implementation makes sure the timing leakage is small. The CoverUp downlink and uplink rate of our prototype is between 10 and 50 Kbit/s, depending on the tolerable bandwidth overhead, and the average latency is 3 seconds.
- We experimentally evaluate the timing-leakage of our prototypes by measuring the differences between active and passive users in the network timing-delays (challenge (*iii*)). Using these measurements as a model for the timing leakage, we are able to bound

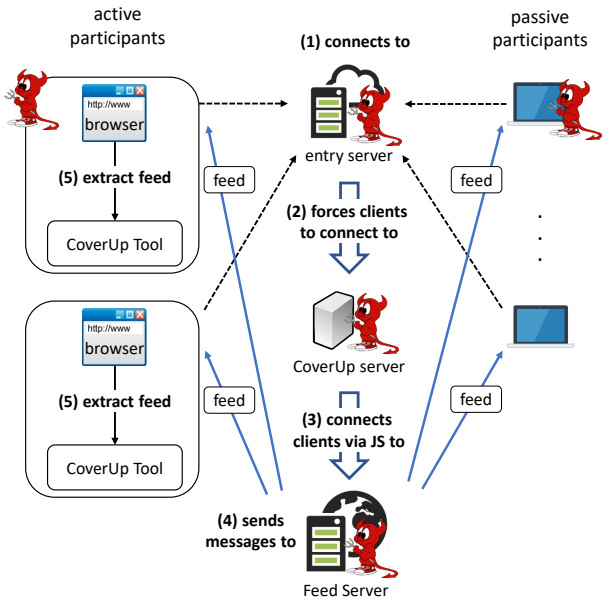

Fig. 1. Main components of CoverUp for CU:Feed. All visitors of an entry server are redirected to the CoverUp server, triggered to send (dummy) requests to the Feed server, and then receive an encoded piece of a uni-directional message feed (4), which is extracted (5) by active participants via the CoverUp-Tool.

an attacker's accuracy after half a year of continual observation[1] with 55%.

- We discuss ethical and selected legal questions w.r.t. the entry server and the passive participants.

2 CoverUp

Passive Participation raises the challenges of how to trigger passive participants to join an ACN (or connect to a feed server) with unintrusive technologies, while rendering the traffic of active and passive participants indistinguishable.

2.1 CU:Feed

CoverUp uses for CU:Feed a lean design that leverages common and widely-used JavaScript functionality of browsers to trigger visitors (the *passive* users) of a cooperating website (the *entry* server) after an informed consent to produce unsuspicious cover traffic for users that are interested in the message feed (the *active* users). This cooperating website (the *entry server*) could be a university, a knowledge, or a news site.

The entry server (Figure 1, Step 1) embeds in its HTML-code an iframe to a dedicated server (the CoverUp *server*) with a different domain. This CoverUp server responds with a JavaScript code snippet (Step 2). This JS snippet triggers the browsers of the visitors (Step 3) to continuously send requests to the *Feed server*, who responds with CU:Feed packets (Step 4), effectively producing cover traffic to and from the Feed server. As the CU:Feed has no strict latency requirements, the browser behavior of active participants can be kept exactly the same, thus avoiding timing leakage. CoverUp stores each dummy packet in the browser's `localStorage` cache, and an active participant uses a previously obtained external application (CoverUp-Tool)[2] to extract these CU:Feed packets (Step 5) from the browser cache. Hence, only this extraction, i.e., each file-read operation, indirectly produces minimal timing leakage.

With regard to the privacy of passive participants, the JS snippet from the CoverUp server is in an isolated context and thus can not learn anything due to the same origin policy. Hence, the CoverUp server only learns when a participant visited the entry server, by the requests.

The CU:Feed could contain controversial content. To deflect potential legal harm to the passive participants, we cryptographically protect them from accidentally storing parts of the CU:Feed on their disc by utilizing an all-or-nothing scheme [62] and only storing a small amount of CU:Feed packets in the local storage. Without actively trying to, passive participants do not have sufficient of information from which any content of the feed could potentially be reconstructed. We discuss the legal importance of this protection in Section 6.

The Feed server does not know which packet has reached a user and in which order. We use error-correcting Fountain Code (see Section 4.2) which enables the assembling of CU:Feed packets in an arbitrary order and with potentially missing packets. The authenticity of the feed can be achieved via signing the feed and assuming a PKI.

1 We assume that a usage pattern of at most 45 times a day, and at most 5 hours in total (see Section 5.6).

2 CoverUp-Tool could be obtained off-the-record or as part of the CU:Feed. There, a small program including explanation could be distributed in clear text and without Fountain-Code which could be extracted from the cache manually. This program assembles the full CoverUp-Tool delivered by the encoded feed.

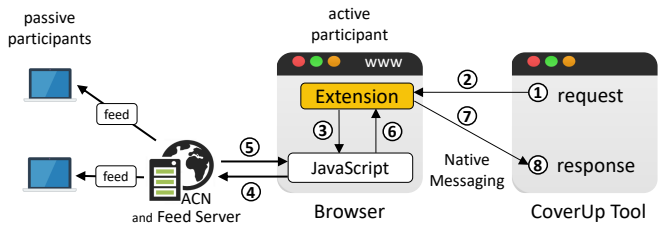

Fig. 2. CU:Messenger in combination with the CU:Feed. Once the JS snippet has been received, all participants request CU:Feed packets. An active CU:Messenger user can use these requests to inject custom requests to the ACN by a browser extension. To render the traffic from passive and active users indistinguishable, we use TLS encryption (yellow arrows) at (Step 4), (Step 5) and all connections of passive participants in contrast to CU:Feed. For CU:Messenger the dummy messages do not need to contain a feed; they can also purely contain garbage.

Trust assumptions and attacker capabilities. The CU:Feed tolerates a global network-level active attacker that controls all parts of the system except the active participants operating system and its running applications, as the only difference between active and passive participants is the local cache-reading CoverUp-Tool. This attacker is active, so he can modify, drop or delay any number of messages. As we focus on guaranteed anonymity and not on integrity, CoverUp is not censorship resistant as it cannot protect from denial of service.

Tackling the bootstrapping problem of ACNs While CoverUp also provides participation time hiding and thereby deniability without an ACN, sufficiently strongs ACNs can benefit from the additional cover-traffic that CU:Feed produces. CoverUp (active and passive) users would use the ACN to connect to a server that broadcasts information (e.g., podcasts or RSS-feeds). ACNs that resistant to traffic correlation, such as Vuvuzela [67] or DISSENT [43], would benefit from the additional (legitimate) traffic that CoverUp users would produce. Hence, CoverUp can help with the bootstrapping problem of an ACN.

2.2 CU:Messenger

We extend the uni-directional CU:Feed into a deniable entry point for bi-directional point-to-point communication with an ACN, the *CU:Messenger*. For the uplink communication, our bi-directional channel requires the active participant to additionally install a browser extension.

The protocol is almost the same as the CU:Feed. While passive participants only transmit and receive CU:Feed messages, also active participants transmit dummy messages unless they need to transmit content (Figure 2, Step 1). In those cases, they use native messaging to connect from the CoverUp-Tool to the browser extension (Step 2). The browser extension then replaces a CU:Feed request with a real message content (Step 3). All messages (Step 4) are encrypted by TLS (using randomized encryption), of the same size, and are transmitted at regular time intervals. This renders dummy messages and real messages indistinguishable for the network-level adversary. Upon receiving the encrypted message (Step 5), the browser extension records it and hands it over to the CoverUp-Tool which decrypts the content (Step 6) and sends it via native messaging (Step 7) to the CoverUp-Tool (Step 8).

CU:Messenger trust assumptions. The CU:Messenger can tolerate a global network-level attacker and a malicious entry server. However, the integrity of the JS snippet has to be guaranteed; otherwise an attacker can inject malicious JS code which can detect active participants (e.g., by testing for the existence of the extension). To enable the browser extension to check the integrity of the JavaScript snippet with minimal timing leakage, we trust the CoverUp server in this application and the browser extension simply checks whether the origin of the JavaScript code snippet is as expected. If the check fails, the browser extension does collect or hijack any packets. We require the ACN to provide strong anonymity [3] since otherwise the requests of passive participants to the Feed server can be distinguished from the CU:Messenger connections.

It is possible to tolerate malicious CoverUp server if we check the integrity of the JavaScript code byte for byte. Such a check would, however, produce a significant timing leakage and is hence not an option. The entry server can be malicious as it can not distinguish between the active and passive participant. The entry server may redirect to a malicious CoverUp server but in that case, the browser extension remains inactive as

3 We assume that the ACN satisfies sender and receiver anonymity in the sense of AnoA [31] with negligible adversary-advantage.

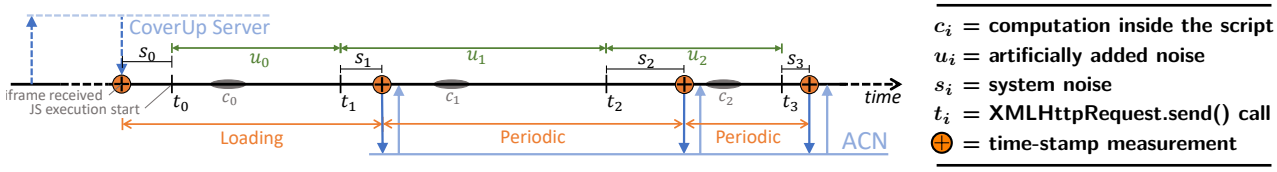

Fig. 3. The timeline of active and passive participants in their browser, starting at a request for a JavaScript code snippet from the CoverUp Server in an iframe. The code is executed and makes continuous requests to the ACN. The attacker can measure network timestamps of the requests (⊕). To decrease the leakage s_i of the system or browser internals, we add randomly chosen delays u_i to the sending times t_i. There are two main sources of leakage: the set-up of the iframe context (Loading) and the interval between the consecutive requests (Periodic). Any comprehensive computation c_i inside the script or by the browser extension (for active users) is done between the sending intervals when all components are idle.

it strictly checks for the proper CoverUp server. We assume that the target ACN acts as an 'honest' ACN or that it can be implemented in a distributed fashion so not to leak information about the communication between the users. We assume that the ACN implements a secure channel, in particular, that it provides confidentially and authenticity between the two communicating parties. The anonymity guarantees of the CU:Messenger can, of course, only be as good as the anonymity guarantees of the ACN. We assume that the ACN satisfies sender and receiver anonymity in the sense of AnoA [31]. We consider this to be orthogonal to our work in this paper.

2.2.1 Requirements on the ACN

Does CoverUp work with any application of ACNs? No, CoverUp increases privacy if the receiving party collaborates and treats dummy messages as normal messages, e.g., a messenger application could do that. The dummies could be used to distribute a feed.

For the ACN itself, we assume sender and receiver anonymity in the sense of AnoA [31]. We additionally need the ACN to constitute a secure channel, in particular, it has to provide confidentially and authenticity between the two communicating parties.

We require that the ACN handles client-side dummy traffic in a manner that is indistinguishable from real client traffic for a global network-level attacker, while continuously delivering to all parties some messages (potentially dummy messages) according to a fixed user-behavior distribution (e.g., at a constant rate). In particular, CoverUp is not compatible with Tor because traffic correlation attacks enable a global network-level attacker to realize whether a client-side produced onion is delivered to a recipient. In our experimental evaluation, we model the ACN in the experiments as a central mix to abstract away from ACN-specific effects.

Best suited for CoverUp are mix nets that are based on flushing algorithms, such as threshold mix-nets, timed, threshold pool, timed pool, timed dynamic pool, stop-and-go, and binomial mix-nets. CoverUp can, in particular in combination with a messenger application improve the anonymity set, which in turn could help to bootstrap a user base. Additionally, CoverUp provides the property that an active participant can deny the intention to participate in the ACN.

Apart from these mix flushing algorithm, there exist several types of mix based on transport privacy scheme. Here we list some well known transport-private ACNs.

Store-and-Forward. This ACN requires some transfer delay and storage at the intermediate server (e.g., Email/XMPP). CoverUp can be plugged to such ACNs as long as secure communication is ensured.

Onion routing. While low-latency onion routing protocols, such as Tor, may not be good a candidate for CoverUp, as Tor is susceptible to traffic correlation attacker and requires a low latency. Also, CoverUp uses a fixed user-behavior distribution (e.g., constant rate) between the users and the ACN. This can drastically increase the bandwidth requirement of Tor making it infeasible for CoverUp.

DC-Nets. DC-nets executes in rounds. In every round, a participant can either submit a secret message or not submit anything. CoverUp would be suitable for DC-net as all the participants can submit the messages in its round. The active participants would submit legitimate encrypted message while the passive participants submit random bit string. DC-nets can experience higher latency due to increase number of participants (due to passive users).

Message broadcast. Message broadcast protocols are suitable for CoverUp. Here all the participants re-

ceive messages from others but only decrypt a specific part of the broadcast if the message is intended for him.

PIR. CoverUp is compatible with PIR based scheme but will increase the computational complexity much fold due to the number of passive participants.

2.3 Timing leakage

Involving passive participants necessarily leads to timing leakage, as long as active participants have to use additional applications. To read the feed, a small external application (CoverUp-Tool) needs to extract the CU:Feed packets from the browser cache. This application shares system-wide computation resources with the browser and influences its computation time, even though it does not directly interact with the browser. The browser's altered reaction time constitutes timing leakage that can be observed by a global network-attacker. The CU:Messenger requires direct interaction with the browser, which leads to even more significant timing leakage.

This leakage cannot be countered by introducing a delay that extends the execution time of active and passive parties to the same time, since a JavaScript program on a passive participant's machine has no way of getting a precise value of the delay and of enforcing a precise delay. We introduce random delays and show in Section 5 that these random delays significantly reduce the timing leakage. To limit the amplification of the timing leakage, we additionally limit the number of requests for which the browser extension (of an active participant) is active. This limits the risk of malicious entry servers triggering excessing amounts of page-loads, which would otherwise enable an attacker to dramatically increase the number of observations and in turn increase the timing leakage.

Figure 3 illustrates the timeline of how messages are sent, received and processed in the browser by CoverUp and which observations a network-layer attacker can perform. The system delay s_i in this figure refers to a system's computation time (including delays caused by the OS, the browser, and the network card). The figure also shows when the noise is added and, in the case of the CU:Messenger, when the communication with the extension takes place. For the quantification of the timing leakage, we simplify the observations and partition the execution into two types of measurement an attacker can perform and which we assume to be independent of each other. These intervals are the same for the CU:Messenger and the CU:Feed. *Loading measurements* denote the time between the reception of the JavaScript snippet from the CoverUp server and the first outgoing request to the ACN or Feed server. *Periodic measurements* catch the time between subsequent CoverUp requests to the ACN or Feed server.

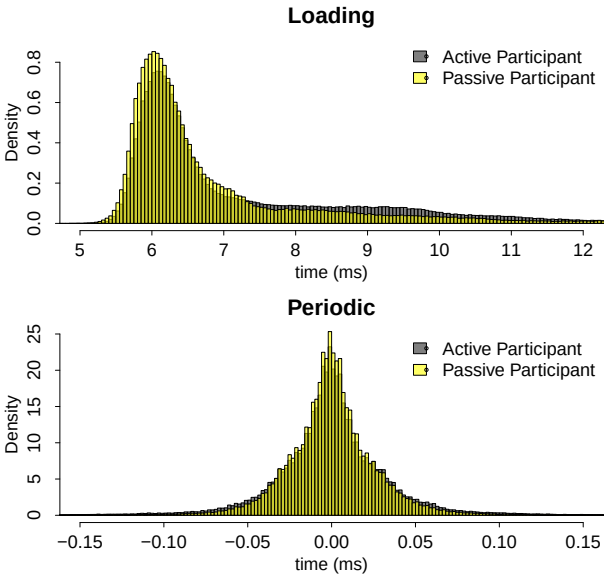

Fig. 4. Distribution of timing delays (without additional noise) of Loading and Periodic measurements run on Linux. Each of the graphs overlays the timing distributions of active and passive participants.

We use this partitioning of the execution to construct a histogram of Loading measurements and one of durations of Periodic measurements. As an illustration, Figure 4 shows the distributions of timing delays from active and passive participants both for Loading and for Periodic measurements in the CU:Messenger case.

3 Privacy implications

This section discusses the privacy-benefits and the limits of CoverUp and the privacy-related drawbacks of deploying CoverUp. We also discuss how we capture timing leakage.

3.1 Potential sources of privacy leakage

We took great care to carefully design the system in a way that the privacy leakage is minimal. We minimize the leakage from the traffic with the ACN (see Section 3.1.1), in particular the timing leakage, which we separately quantify in Section 5. Nevertheless, using CoverUp as an entry point for an ACN can in-

fluence the behavior of active participants toward the entry server, which we discuss in Section 3.1.2. Additionally, a curious CoverUp server can learn the time at which passive participants visit the entry server, which we discuss in Section 3.1.3. Potentially, browser profiling methods can be used to learn whether a particular extension is installed [38]. However, measuring these effects is out of the scope of this work.

3.1.1 Participation deniability

The main focus of this work is to quantify the accuracy with which active participations can be distinguished from passive participants, as long as their browsing behavior on the entry server is the same (see Section 3.1.2 for more). We introduce a quantitative notion of acc-*participation deniability* that captures that an attacker can distinguish the observable communication of any active participant from any passive one only with a certain (low) accuracy, as long as their behavior on the entry server is the same. This section gives a high-level idea of our quantitative notion: participation deniability. Section 9.1 offers a more precise explanation. We over-approximation any potential prior knowledge by enabling the attacker to choose a browsing behavior on the entry server and can determine what the active participant would do in the ACN. Additionally, we grant the attacker perfect knowledge of the timing delay distributions for the pair of the active and the passive users.

For the analysis of the *CU:Feed*, the attacker controls the network and all parties except for the user's machine. For the analysis of the *CU:Messenger*, the attacker controls the network and all parties except for the CoverUp server and the ACN (which typically means that only a fraction of the ACN is compromised). As long as the behavior on the entry server is the same, it can be shown that CoverUp only leaks timing delays that are due to the browser extension and the CoverUp-Tool. We stress that our attacker model explicitly excludes side channels like access to the power consumption, noise generation, or any radiation leaks (e.g., heat) of a trusted client. The technical report [9] contains formal definitions, a precise treatment of the attacker model, and proves the statement above.

This timing leakage is inherent to the concept of Passive Participation, as CoverUp has very little control over the passive participants' machines. Hence, we use a quantitative variant of the indistinguishability notion. More precisely, we say that a pair of protocols has acc-*participation deniability* if a distinguisher that interacts with one of these protocols (which one is randomly decided) cannot decide (in the sense of classify) with better than acc accuracy with which protocol it is interacting. Let us consider a precise definition for the simplified non-interactive case. We choose a definition that is inspired by the total variation, which can be proven to be the advantage of the optimal unbounded attacker. We consider unbounded attackers, since we abstract away from any cryptographic leakage that could be broken by an unbounded attacker and solely restrict ourselves to timing leakage. Formally, we assume idealized version of cryptography, see Section 9. For two X, Y discrete distributions over a finite domain with a joint domain Ω, the *accuracy bound* $\mathtt{acc}$ of X and Y is defined as follows[4]

$$\mathtt{acc}(X,Y) := \frac{1}{4}\sum_{a\in\Omega}(|p_X(a) - p_Y(a)|) + 0.5$$

In other words, we provide a bound on the attacker's distinguishability accuracy. Specifically, n continual collected observations can be modeled by considering $\mathtt{acc}_{n,\{X-Y\}} := \mathtt{acc}(X^n, Y^n)$ for the product distribution X^n and Y^n. Another way of looking at the definition is that it requires the classification accuracy of the interaction with active versus with passive participants to be bounded by a number close to 50%.[5]

3.1.2 Privacy leakage due to behavior changes

The usage of CoverUp may unconsciously influence the behavior of active participants, e.g. if active users spend more time on a specific entry server in order to use CoverUp thus significantly reduce the anonymity set. This problem is inherent to our approach and requires a thorough study. As a countermeasure, one could include a 10-minutes delay before the connection to the ACN would start. Recent studies show that the average visiting time of e-commerce website is between 8.7 and 10.3 minutes (2016 Q1) [2, 16]. Hence, those users that keep tabs open in the background would also with-

4 This bound acc can be translated to the total variation (also known as statistical distance) δ as follows: $\delta = 2 * (\mathtt{acc} - 0.5)$.

5 In literature, the classification accuracy is computed as the mean over several runs, which leads to a non-zero variance. Thus, this empirical accuracy can be above our bound which is defined over all possible traces of collected observations. So, the precise statement is that, with an increasing number of runs, the classification accuracy converges towards a value that is bounded by our accuracy bound acc.

out CoverUp after 10 minutes not browse on the entry server anymore.

A malicious network-level attacker or a malicious CoverUp server could mostly block CoverUp packets to provoke a user to connect longer to the entry server than it would normally do. A user might try to compensate by staying longer connected to the entry server, thereby changing its entry-server usage behavior. For users that remain patient and do not change their behavior in such a situation, this attack reduces to a plain Denial-of-Service attack.

3.1.3 Browsing time of passive participants

Inactive users of CoverUp potentially reveal their browsing behavior to the CoverUp server, as a malicious CoverUp server can read HTTP header's referrer field. This leakage is inherent in our approach to use an entry server and to utilize passive participants to produce cover traffic. While this leakage exists, we would like to put it into perspective. Many popular websites already leak this information to other services, such as advertisement networks or external analytic tools, such as Google Analytics.

A curious CoverUp server would be able to learn the time at which a person visits the entry server. We define the scope of the iFrame to be the entire domain. In this way, an honest-but-curious CoverUp server only learns the time at which a user visits the entry server but not every single time it opens a webpage. This can also achieve by letting the entry server keep track of which user already has a JS file.

3.2 Privacy effects

The indistinguishability of the ACN-traffic of active and passive participants enables hiding the act of participating in an ACN, as long as there are no suspicious changes in the active participants' behavior towards the entry server (see Section 3.1.2).

3.2.1 Hiding the act of participating in an ACN

Let us consider the cases where the browsing behavior towards the entry server does not change, as discussed in Section 3.1.2, e.g., if an artificial delay of 10 minutes is introduced before anything is requested via CoverUp and the connection is only used when the tab is in the background. In those cases, using CoverUp as an entry point to an ACN can provide deniability for the act of participating in an ACN. Unsuspicious passive participants generate a high amount of cover traffic that is indistinguishable from active participants.

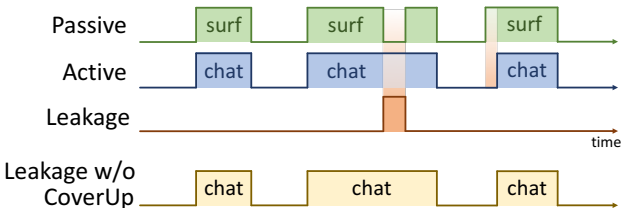

Fig. 5. Simplified depiction of how CoverUp can hiding the act of chatting. The x-axis is the time, and the y-axis is whether at that time surfing or chatting behavior is expected (which assumes an strong attacker that has strong background knowledge). Only the chatting activity which is not covered by *expected* surfing behaviour creates leakage. If a user chats only when he surfs, this kind of leakage is zero. Else if the chatting activity is not completely covered, CoverUp still reduces the leakage, prolongs detection time, and increases privacy compared to absence of CoverUp.

Figure 5 illustrates this property by concentrating on the participation time alone. While an traditional perfect ACN leaks the participation time, using CoverUp as an entry point can hide large parts and even all of the participation time. Let us assume an attacker that has perfect knowledge about a user's surfing and chatting behavior, as depicted in Figure 5. Even if the urge to chat slightly influences the surfing behavior (marked in Leakage), the leakage is much smaller than directly communicating to an ACN directly without CoverUp (marked in Leakage w/o CoverUp).

If the chatting urge between two participants is statistically independent, i.e. they only chat if they are randomly on-line at the same time, and they do not adapt their surfing behavior to that urge, then they are indistinguishable from any passive participant. The wider CoverUp is deployed, the higher is the probability that they are simultaneously available.

3.2.2 Bootstraping the ACN

CoverUp increases the anonymity set with the set of passive participants, for the CU:Feed (Section 2.1), the JavaScript code snippet would visit the Feed server via an ACN. Note that both the active and passive participants execute the identical protocol (only change being the active users uses the CoverUp-Tool to decode the

feed). This makes the active and passive users indistinguishable to the global network level adversary.

4 CoverUp's implementation

This section describes the CoverUp prototype implementation and presents its performance.

4.1 Preliminaries

In this section we describe existing tools and techniques that have been used in our proposed system CoverUp.

4.1.1 Fountain Code

Fountain codes [58, 65] are a class of forward error correction (FEC) codes with the following properties

- Arbitrary sequence of encoding symbols can be generated form a given set of source symbols i.e., input data.
- Original source symbols can be recovered from any subset of encoding symbols with size more than a threshold value T.
- Encoding symbols can be delivered regardless of specific order.
- Fountain codes does not show fixed code rate.

In this paper, we have used a bit-wise XOR ($\oplus$) based fountain code with error detection mechanism.

In a simple analogy, one can consider an empty glass for water. A fountain emits the input data encoded in a large amount of droplets in a steady stream. Anyone can collect them in a glass alternately and if one thinks the glass is filled enough, one may try to assemble the data from the water (data stored in the glass). If the amount of droplets is insufficient to reassemble the data, one has to wait longer to collect more droplets and retries later.

Our specific fountain code implementation is not optimal. There exists efficient fountain codes such as *Raptor* [64] in the literature but most of them are protected by intellectual property rights.

4.1.2 All-or-nothing transformation

All-or-nothing transformation is an encryption mode in which the data only can be decrypted if all the encrypted data is known. More precisely: "An AONT is an un-keyed, invertible, randomized transformation, with the property that it is hard to invert unless all of the output is known."[36].

We modified the *all-or-nothing scheme* proposed by Rivest [62] which encrypts all data with a symmetric key cryptography algorithm (in our implementation, we use AES-128 [44]) in Cipher Block Chaining (CBC) mode and appends a new block in which the encryption key is XOR'ed ($\oplus$) with the 128 bit truncated SHA-256 hashes of all the encrypted blocks. This guarantees that one needs all encrypted data (or at least its hash) to extract the decryption key from last block.

1. Input message block: $m_1, m_2, \ldots, m_n$
2. Chose random key $\mathcal{K} \xleftarrow{R} \{0,1\}^{128}$ for AES-128.
3. Compute output text sequence $m'_1, m'_2, \ldots, m'_n, m'_{\text{key}}$ as follows:
   - Let $m'_i = Enc(\mathcal{K}, m_i)\ \forall\ i \in 1, \ldots, n$ with CBC mode.
   - Let $m'_{\text{key}} = \mathcal{K} \oplus h_1 \oplus h_2 \oplus \ldots \oplus h_n$ where $h_i = \mathcal{H}_i[1, \ldots, 128]; \mathcal{H}_i =$SHA-256$(m_i)\ \forall i \in 1, \ldots, n$
   - Send $m' = m'_1 || \ldots || m'_n || m'_{\text{key}}$

The receiver can recover the key $\mathcal{K}$ only after receiving all message blocks. He executes the following steps

- $\mathcal{K} = m'_{\text{key}} \oplus h_1 \oplus h_2 \oplus \ldots \oplus h_n$.
- $m_i = Dec(\mathcal{K}, m'_i)\ \forall\ i \in 1, \ldots, n$.

4.2 Prototype implementation

We implemented a prototype and made it available under http://coverup.tech. The CoverUp implementation consists of five components: a CoverUp server, a dummy mix server that acts like a message relay & broadcaster, an external application (CoverUp-Tool), a browser extension, and a short JS code snippet. The CoverUp server and the mix server is implemented as a Java Servlet running on an Apache Tomcat web server. The external application is written in Java. The browser extension is implemented in Google Chrome browser using the JS WebExtensions API. The JS code served by the entry server is kept at the CoverUp server. The CoverUp-Tool and the server implementation consists of about 14 KLoC and the browser extension of about 200 LoC.

We make the following four assumptions about the browser, which are in line with Chrome's explicitly stated security policies. 1. iframes are isolated, which we need for the code integrity of CoverUp's JS snippet. The parent page of the iframe cannot modify the iframe if the iframe is originated (domain) from a source

other than the parent [7]. 2. a JS code cannot read from or write to another context of a different domain source without its consent. 3. the JS code can write a small amount of data to the browser's localStorage cache and this cache cannot be accessed by another JS code which originates from a different origin. This property is known as the "same-origin-policy" [10], and all modern browsers claim to enforce it.

The system is parametric in the payload size and the request rate. The payload size denotes the size of the HTTP request and response packets which are exchanged between the users (both active and passive) and the mix server. We vary this size from 3.75 to 18.75 kbytes. The request rate denotes how frequently the JS snippet delivered from the CoverUp server executes a request to the mix server. Our implementation uses a rate of 3 seconds. Section 5 evaluates our choices for these system parameters. Increasing the payload increases the traffic overhead, in particular of passive user's, and reducing the request rate reduces the latency but decreases the privacy (see Section 5). Hence, there is a natural trade-off between the latency and privacy and the amount of traffic overhead cause and throughput of the system.

For the CU:Feed, all the users of the entry server receive identical broadcast content which is encoded with a fountain code [58]. Such encoding ensures that any out of order threshold amount of broadcast packet can recover the data successfully. Our prototype implementation uses a `XOR` based fountain code (details in [9]). The JS snippet served by the CoverUp server stores the fountain pieces in the cache database file located on the mass storage (known as browser `localstorage`). The CoverUp-Tool collects and assembles the fountain pieces from the `localstorage`. Our implementation also employs an All-or-Nothing-Encryption scheme (one similar to [62]) which ensures that one needs threshold amount of pieces of the fountain (i.e. the entire source data) to decrypt it. The JS snippet only keeps one fountain piece in the `localstorage` to ensure that the passive users do not have any sensitive content on their disk in decipherable form.

CU:Messenger. We extend the uni-directional CU:Feed to the bi-directional CU:Messenger (recall Section 2.2), which provides an entry point fo those ACNs characterized in 2.2.1.

The implementation of the CoverUp messaging protocol involves indexing the messages as POP (post office protocol [26]) where the indexing is done by public addresses of the clients. This public address is derived from the *curve25519*[34] public keys (first 48 bits of the hashed public key). Additionally, the mix server indexes these public addresses by the SSL identifier of a specific HTTP request. This helps the mix server to uniquely identify a sender/receiver from the incoming connection request. The CU:Messenger application assumes that the user added all long-term public keys of all his trusted peers. For the cryptographic protection for the messages, the application computes a shared secret (using Diffie-Hellman key exchange) from the long-term key pairs. The current prototype of CoverUp does not provide forward secrecy, but one can easily integrate such feature into the messenger. Whenever a new message arrives from a source address, the mix server keeps the message to the index of the destination address. When a request arrives at the destination address, the mix server delivers the message as the response and removes the message from the previously kept index location.

4.3 CoverUp performance

We estimate CoverUp's overhead, latency, and throughput to demonstrate that it can perform reasonably well in a real-world scenario, is feasible for deployment in large scale and does not incur an intolerable overhead. CoverUp has three adjustable system parameters: request payload size, response payload size and the average request frequency, which is the average requesting rate for CU:Feed packets after adding artificial noise. A lower request frequency leaves room for more artificial noise and thus increases privacy.

In our prototype implementation, the request/response payload size is in the range of 3.75 KB to 18.75 KB. We send a request every 3 seconds in average for CU:Messenger and CU:Feed. Due to the browser extension's timing leakage (see Figure 4), we noise the individual sending delay to reduce leakage (see Section 5).

Computational overhead. The computational overhead of CoverUp's JS executed int the Browser is negligible. Our implementation of the CoverUp-Tool takes around 50 MB of main memory and less than 1% CPU time. Similarly, installation of the CoverUp browser extension incurs an almost unnoticeable amount of memory and CPU consumption.

Traffic overhead. The traffic overhead of CU:Feed and CU:Messenger are identical, as they are indistinguishable by design. The entry server's overhead is minimal: Only the size of the iframe tag in its HTML code. The passive participants' traffic overhead depends on

Type	Throughput	Latency
CU:Messenger and CU:Feed	10 to 50 Kbits/s	$3s + RTT$

Table 1. CoverUp's throughput for different choices of packet length, ranging from 3.7KB to 18.75KB. We assume that the random delays are chosen such that in the expected case every 3 seconds, a packet is sent. RTT denotes to the network roundtrip time from the user to the ACN and back.

the system parameters. To find suitable values for the system parameters, we looked at the Alexa top 15 news sites, in particular since the privacy improvements of CoverUp's Passive Participation approach depends on the entry server's regular number of visitors. The average main-page load-size of the Alexa Top 15 news sites is around 2.2 MB and will grow in near future. A few examples are CNN with 5.6 MB, NYTimes with 2.4 MB, HuffingtonPost with 6.1 MB, TheGuardian with 1.8 MB, Forbes with 5.5 MB, BBC with 1.1 MB and Reddit with 0.8 MB.

CoverUp is parametric in the packet size. Once fixed, the traffic overhead for the passive users is proportional to this packet length. We generously assume a passive participant that has a daily connected to the entry server for 5 hours each day. This participants would have 22MB ($\sim 5 \cdot 60 \cdot 60s \cdot \frac{1}{3s} \cdot 3.75$KB) to 110 MB ($\sim 5 \cdot 60 \cdot 60s \cdot \frac{1}{3s} \cdot 18.75$KB) of data overhead per day and 660MB ($= 30 \cdot 22$ MB) to 3.3 GB ($= 30 \cdot 110$MB) per month. For landline data flat-rates (i.e., for non-mobile visitors), 22 MB is not significant, e.g., in comparison to the traffic caused by streaming videos. We envision a deployment of CoverUp not to include mobile users. But it may be possible in near future due to the increased bandwidth of the mobile networks. Section 6 further discusses the ethical aspects of using the passive participants' resources.

Latency & throughput. We evaluate the performance of CoverUp for the duration that a tab is opened since the usage of CoverUp is bound to the visiting patterns of passive participants towards the entry server's sites. Depending on the service that the entry server offers, it might not be common to keep the tab open for a long time or to visit the site more than a few times a day. For the performance evaluation, we generously assume that each user is connected for 5 hours per day to the entry server, which is achieved by letting the tab open in the background. Table 1 illustrates the throughput and the latency during a session. CoverUp achieves 10 to 50 Kbits/s (by varying the fixed length packet size from 3.75 to 18.75 KB) throughput and around 3 seconds of average delay between consecutive messages. As the future size of websites will grow, CoverUp's data usage can be adapted to deliver a better performance. 3 second average delay is practical to execute messenger-like services. Section 5 explains our choice for the delays.

5 Timing leakage experiments

How much privacy does CoverUp provide? To answer this question, Section 5.1 first describes our estimator for the classification-accuracy, Section 5.2 the experimental set up, and Section 5.3 our treatment of outliers. After fixing a usage pattern in Section 5.4, Section 5.5 explains the noise-adding process and presents the results: First, how much expected latency leads to which privacy estimates? And second, how does the degree of privacy changes over time? Section 5.6 discusses the limits of our evaluation.

5.1 Estimating the classification-accuracy

Our goal is to give an upper bound on the classification-accuracy for the task of distinguishing active and passive participants. This section explains the estimators that we use. We assume that the dominant part of the timing leakage will be visible from two kinds of measurements: *Loading* and *Periodic* measurements depicted in Figure 3. In Loading measurement, we force the iframe to refresh on the entry server page in the browser. In the corresponding TCP dump, we measure the timing difference between the response of the initial iframe HTML source request and its first ("forced") request to the mix server. This forces Loading the extension's content script and thus captures any distinguishing feature (any timing delay added by the existence of the browser extension) produced by the extension. The Periodic measurement models the scenario where the active and passive participants load the iframe once, followed by entry server served JavaScript generated request to the mix server and one response from the same. In the network traffic dump, we look for the timing difference for two contiguous CU:Feed requests from the browser. Section 5.6 discusses the choice to concentrate on these measurements. We are going to compare the timing measurement distributions of the passive participant against the distribution of the active participant.

As an estimator for the classification-accuracy, we use the accuracy $\texttt{acc}_{n,type}$ defined in section 3.1.1. This accuracy depicts the ratio of cases where an attacker classifies a participant correctly as active or passive after n measurement samples collected while having perfect knowledge of the active's and passive's timing measurement distributions of type $type \in \{loading, periodic\}$. If we assume that each request time is drawn from an independent distribution (see Section 5.6 for a discussion), we can bound the classification-accuracy $\texttt{acc}_n$ using Ratiobuckets: a recently introduced and publicly available numerical tool that computes a provable upper bound for the total variation of a given pair of discrete distributions [60]. We run the Ratiobuckets-tool, on the pair of measurements for Loading and Periodic observations for Linux and Windows. Since the Ratiobuckets-tool is most precise when $n = 2^x$ for some x, the number of compositions that we use are powers of 2. Using standard composition results (see Appendix Lemma 1), we can then bound the classification-accuracy of CoverUp with $\texttt{total}_{n,m} := [\texttt{acc}_{n,loading} - 0.5] + [\texttt{acc}_{m,periodic} - 0.5] + 0.5$, after attacker that makes n Loading observations and m Periodic observations for either Linux or Windows. Here we explicitly use the assumptions that Periodic and Loading measurements are independent.

Definition 1 (Total variation over finite domain)**.** *Let X, Y be two discrete distributions over a finite domain with a joint domain Ω. Then, the* total variation *d of X and Y is $d(X,Y) := \frac{1}{2}\sum_{a\in\Omega}(|p_X(a) - p_Y(a)|)$.*

Lemma 1. *Let X_l, X_p be the Loading, respectively the Periodic, measurement distribution of the passive user and Y_l, Y_p the Loading respectively the Periodic measurement distribution of the active user, all with a joint Domain Ω. Let further be δ_l be the total variation between X_l and let Y_l and δ_p be the total variation between X_p and Y_p. A distribution with the superscript n or m denotes the resulting distribution after n or m draws of the originating distribution. Then, for all Turing machines A, if all the measurement samples are independent (**AI**), Loading and Periodic measurements are independent (**AII**), and the measured distributions represent the accurate underlying distributions (**AIII**),*

$$\begin{aligned}&|\Pr[b = 1 : b \leftarrow A(w_l, w_p), w_l \leftarrow X_l^n, w_p \leftarrow X_p^m]\\ &- \Pr[b = 1 : b \leftarrow A(w_l, w_p), w_l \leftarrow Y_l^n, w_p \leftarrow Y_p^m]|\\ &\leq n \cdot \delta_l + m \cdot \delta_p\end{aligned}$$

Let $w \xleftarrow{n} X$ denote n independent draws from a distribution X. Let $\Pr[w \leftarrow X] = \Pr[b = 1 : b \leftarrow A(w), w \leftarrow X]$ and $\Pr[w_l \leftarrow X_l \bowtie w_p \leftarrow X_p] = |\Pr[b = 1 : b \leftarrow A(w_l, w_p), w_l \leftarrow X_l, w_p \leftarrow X_p]$. We conclude:

$$\begin{aligned}&|\Pr[b = 1 : b \leftarrow A(w_l, w_p), w_l \leftarrow X_l^n, w_p \leftarrow X_p^m]\\ &\quad - \Pr[b = 1 : b \leftarrow A(w_l, w_p), w_l \leftarrow Y_l^n, w_p \leftarrow Y_p^m]|\\ &= |\Pr[w_l \xleftarrow{n} X_l \bowtie w_p \xleftarrow{m} X_p] - \Pr[w_l \xleftarrow{n} Y_l \bowtie w_p \xleftarrow{m} Y_p]|\\ &\overset{\text{AI}}{\leq} |\Pr[w_l \xleftarrow{n} X_l \vee w_p \xleftarrow{m} X_p] - \Pr[w_l \xleftarrow{n} Y_l \vee w_p \xleftarrow{m} Y_p]|\end{aligned}$$

$$\begin{aligned}&= |\Pr[w_l \xleftarrow{n} X_l] + \Pr[w_p \xleftarrow{m} X_p]\\ &\quad - (\Pr[w_l \xleftarrow{n} Y_l] + \Pr[w_p \xleftarrow{m} Y_p])|\\ &= |\Pr[w_l \xleftarrow{n} X_l] - \Pr[w_l \xleftarrow{n} Y_l]\\ &\quad + (\Pr[w_p \xleftarrow{m} X_p] - \Pr[w_p \xleftarrow{m} Y_p])|\\ &\leq |\Pr[w_l \xleftarrow{n} X_l] - \Pr[w_l \xleftarrow{n} Y_l]|\\ &\quad + |\Pr[w_p \xleftarrow{m} X_p] - \Pr[w_p \xleftarrow{m} Y_p]|\\ &\overset{\text{AII}}{\leq} n \cdot |\Pr[w_l \xleftarrow{1} X_l] - \Pr[w_l \xleftarrow{1} Y_l]|\\ &\quad + m \cdot |\Pr[w_p \xleftarrow{1} X_p] - \Pr[w_p \xleftarrow{1} Y_p]|\\ &\overset{\text{AIII}}{\leq} n \cdot \delta_l + m \cdot \delta_p\end{aligned}$$

5.2 Experimental set-up

To simulate realistic scenarios, we set up the passive and both kinds (CU:Messenger and CU:Feed) of active participants on 12 identical systems running Windows 10 and Ubuntu 16.04 (both x86-64 and in dual-boot configuration) equipped with an Intel Core i5-2400 3.1 GHz CPU and 8 GB of main memory. Additionally, the CoverUp and a dummy implementation of a mix server run as an Apache Tomcat web server instance (works as an ACN, a messenger relay and a Feed server) on a separate machine in the same subnet connected by a 10 Gbps switch.

All of the communications between the server and the browser are executed over a local GigaBit Ethernet network. We use *tshark* [28] to capture all such network traffic on the participant's network interface. We compare the distributions of timing traces produced by an active participant to the distribution produced by a passive participant. All the experiments are conducted on these set-ups to investigate the timing leakage of the browser caused by CoverUp's browser extension and external CoverUp-Tool.

Reflecting the attacker model. The attacker model (section 2.1) is reflected in our experiments by taking timing traces from the perspective of the attacker

who has access to all network traffic. Therefore, we captured the traffic on a corresponding network interface. As a network-level attacker can change the TCP flag for timestamps and compel the victim's operating system to add timestamps to the TCP headers [25], we conduct all measurements in the settings where participants, the CoverUp server, and the mix server are in the same GigaBit Ethernet switched network. We calculate this TCP time stamp accuracy by requesting a large number of data chunks from the servers we configured both on Linux and Windows operating system. In Linux and Windows, the time-stamp accuracies are $4000\mu s$ and $400\mu s$ respectively. While requesting, we change the `TSval` field in the TCP header which forces the operating system to put the high-resolution time-stamp on the packet. We then calculate how fast this value changes and the rate of change is the highest resolution that the timestamp field can achieve.

Test modes. We emulate primarily three different user scenarios by using various combinations of the browser extension and the CoverUp-Tool. We use Google Chrome browser v57.0 to run our implemented extension. The extension and the CoverUp-Tool communicate via the native messaging interface (via STDIO). Three different test modes include:

1. *passive user*: Google chrome with no extension and no CoverUp-Tool running.
2. *active CU:Messenger user*: Google Chrome with the extension installed and the CoverUp-Tool running which communicates with the aforementioned browser extension by the native messaging interface.
3. *active CU:Feed user*: Google chrome with no extension and CoverUp-Tool running assembling CU:Feed chunk from the browser `localStorage`.

These are repeated for both Loading and Periodic measurements (Loading and Periodic measurement described in Section 5.1).

Usage profiles. Additionally we constructed one user profile in Linux to understand how the execution of other browsing tabs influences the timing leakage. To demonstrate a simple profile we additionally open another tab in the Google Chrome which is running a high definition ($720p$) video in a loop (see Figure 8).

5.3 Coping with system anomalies

Initial analysis of our timing traces reveals various unpredictable effects. Certain measurement periods produced distributions which were significantly different from the majority of distributions. To cope with this we exclude such distributions.

All the aforementioned test modes (recall Section 5.2) were conducted in measurement-chunks of 6 hours with the identical set-up. To illustrate the problem, one of our 12 machines produces around 10 such recorded measurement-chunks, which in turn sums up to 720 $(6 * 10 * 12)$ hours of data. Each of these 10 seasons per machine provides a histogram (distribution of timing traces). Ideally, all measurement-chunks generated from at least the same machine should look very close, since we are running and re-running the same set-up over and over. However, around 20% (1 to 2 out of 10 measurement-chunks per machine) of all such measurements-chunks across all our measurements produced histograms deviated from the standard shape of the measurements significantly. The differences include additional peaks in the histogram, significant shift of peaks, wider/flatter/distorted peaks etc. During our experiment, we experience some crashes due to some memory leakages and we attribute this anomaly to the running of operating system components, update schedule etc. Due to the sheer complexity of modern operating systems, it is very difficult to isolate the reason. To make the results more robust, we stripped all such anomalies from our data for the final privacy analysis. In particular, we cannot make reliable statements about outliers.

5.4 Usage pattern

We assume a connection pattern to the entry server of at most 45 site-loads (which over-approximates participants that often open and close their browser) and stays connected to the entry server for at most 5 hours per day, which corresponds to a party that leaves the tab of visited pages open in the background for a long time. We assume that the usage pattern of an active participant is identical to the visiting behavior of passive participants (see Section 5.6 for more on this). In the case of partially behavioral overlap, CoverUp stills hides the participation in an ACN or the connection to a Feed server (see Section 3.2.1).

5.5 Adding noise

CoverUp does not send requests at fixed time but rather draws delay noise from a Gaussian distribution

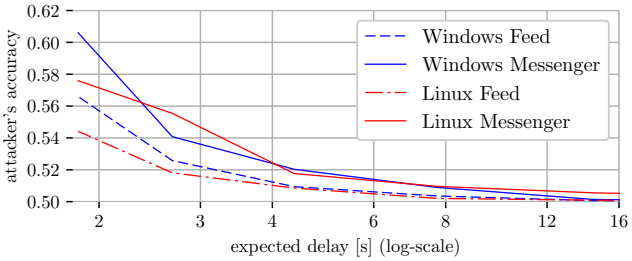

Fig. 6. Latency versus upper bound on classification accuracy for observation of half a year, with at most 5 hours of visiting the entry server (Periodic-observations) and at most 45 time connecting to the entry server (Loading-observations) per day.

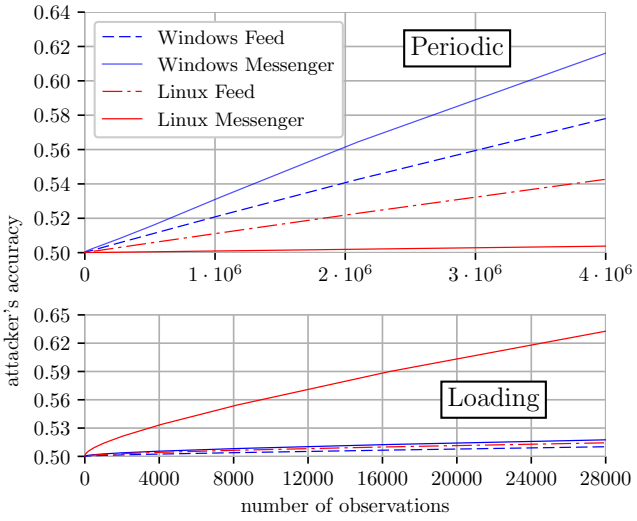

Fig. 7. Number of observations versus upper bound on the classification-accuracy for Periodic and Loading leakage, evolving over numerous observations, with a 3s expected delay.

$\mathcal{N}_{[0,2\mu]}(\mu, \sigma)$ with mean μ and standard-deviation[6] $\sigma = \frac{2}{10}\mu$, restricted to the interval $[0, 2\mu]$, and adds this delay to the minimum delay of one second. The expected delay is therefore $\mathbb{E}\left[1 + \mathcal{N}_{[0,2\mu]}(\mu = \mu, \sigma = \frac{2}{10}\mu)\right] = 1 + \mu$. To simplify and accelerate testing, however, the experiments do not draw this noise. We added noise artificially after the measurements. The effect of separately adding the noise is evaluated in Appendix 5.6.

Latency versus privacy. Fixing the observation time to half a year, Figure 6 plots how $total_{n,m}$ increases with decreasing delays (see Section 5.1). Using the usage pattern from Section 5.4, renders n and m to a function of the observation time. We pick 3s of the expected delay to achieve an overall 55% after 6 months of continual, and highly precise measurements of the user's timing patterns with daily 45 Loading observations and daily 5 hours worth of Periodic observations. We stress that despite the limits of our evaluation, the bounds that we present are highly over-approximated: we assume that the attacker has very precise information about the state of the system such as which processes are running and how they influence measurements. Moreover, the attacker is also capable of conducting high accuracy measurements. Section 5.5 gives a glimpse into the kind of distortion that running another program in parallel brings.

Length of observation vs privacy. The next angle is the length of the observation versus the degree of privacy: Figure 7. We fix the latency to 3s of expected latency and let the number of observation grow, plotting the functions $\texttt{acc}_{n,loading}$ and $\texttt{acc}_{m,periodic}$. This graph lets us study different usage behaviors, e.g., if an e-mail service, such as Google mail or Hotmail deploys CoverUp. In those cases, the sessions are significantly longer than those from visitors from e-commerce web pages. This graph shows that the leakage grows linearly with the number of observations. While Loading needs more time in Linux for the CU:Messenger (presumably because it invokes the extension each time), it produces less Periodic leakage while running.

Distorting effects of concurrent activities. The experiments of which we saw the results so far do not let any other program run in the background. As discussed in Section 5.3, even these kinds of measurements we had a high degree of unpredictability. Figure 8 overlays the histogram of the vanilla experiments (without any other programs running in the background) and experiments where the browser is rendering a 720p video on Linux. The experiments are conducted on Loading events. We can clearly see that the rendering of the video has some impact on the measurement (red line vs. blue line in Figure 8). Hence, it will be hard for an attacker to get such clean measurements like those that we use in our evaluation. This is another reason why we have some confidence that our privacy bounds give a good impression of the degree of privacy that CoverUp can offer, and maybe even provide a significant over-approximation.

Privacy conclusion. For a expected delay of 3 seconds, a half a year's worth of observations correspond to around one million (2^{20}) Periodic observations and around 8192 (2^13) Loading observations . As shown by our graphs, these parameters results for the CU:Messenger and the CU:Feed under Linux and Windows in an attacker's accuracy of $total_{8192,2^{20}} \leq 55\%$, which is only 5% better than pure guessing.

6 There is no specific reason for this σ, but we wanted to prevent hard noise-distribution cut-offs as they increase $\texttt{acc}_n$.

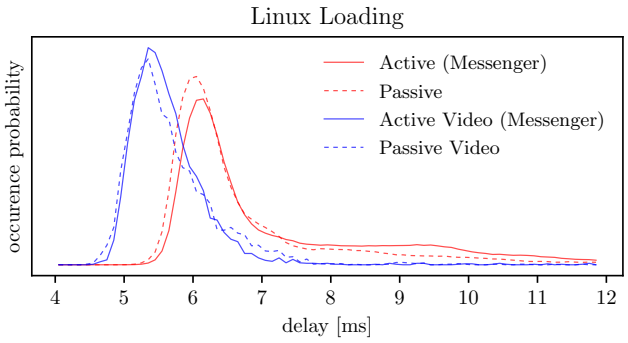

Fig. 8. Different computation loads lead to different timing distributions. In the blue video plots, Google Chrome additionally renders a high definition (720*p*) video in a separate tab. Loading measurement. No randomly chosen delays added.

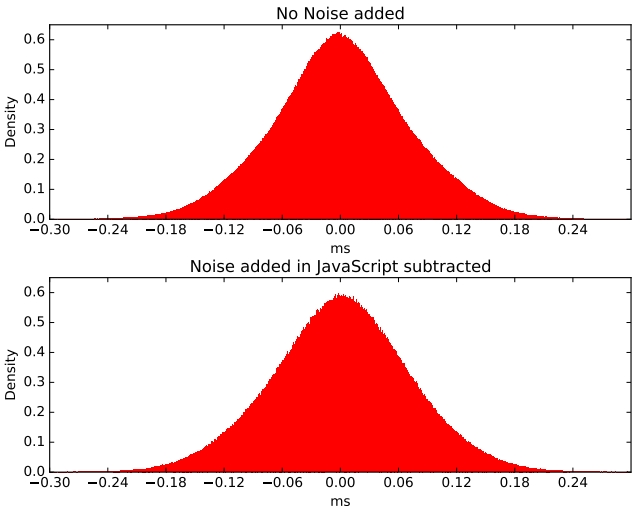

Fig. 9. Statistical Independence using uniform noise: Distance: 1.8%

5.6 Limits of our evaluation

This section discusses the limits of our evaluation and offers an interpretation of our evaluation. While we do not claim that our evaluation offers provable bounds for the privacy of CoverUp, we believe that it captures the dominant part of the leakage of CoverUp and is a good indicator of the privacy that CoverUp offers.

Pairs of requests. We stick to pairs of requests since exploring all possible combinations for a higher number of contiguous requests increases the number of required measurements exponentially. To reduce potential effects from longer sequences of contiguous requests, we incorporate into our recommended delays a minimum of 1s between pairs of requests.

Unnoised measurements. We accelerated our experiments by not adding any additional noise, as we want to evaluate CoverUp with different amounts of noise. During the analysis phase, we introduce noise by computing the convolution of the resulting histograms with ideal uniform noise. To justify this we additionally construct an experiment with two scenarios: one with added artificial noise and another without where we add the artificial noise after the samples are collected.

Recall that we simulated the additional noise by adding it to the measurement result. To justify this procedure, we conducted separate experiments, similar to the periodic scenario, but instead of waiting 1000ms for the next droplet request, we drew in JavaScript a uniformly distributed random number (using `Math.random()`) and expanded it in an affine way such that an interval ranges from 200ms to 1800ms. Additionally, we stored each of the drawn random numbers together with an epoch time stamp. Later in the analysis step, we subtracted the corresponding random number from the network dump measurement. This procedure produced measurements artifacts, caused by the time resolution of our system (which lies slightly under 1us). As we are only interested in the fact whether artificially adding the noise after the experiment is independent of directly adding the additional noise in the experiments, we clustered close histogram bars that are not separated by a significant gap. Figure 9 shows the resulting distribution. The statistical distance of these two distributions is 1.8% which is an acceptable value.

Browser profiling. Potentially, browser profiling methods can be used to learn whether a particular extension is installed (and/or active) or a specific application is running [38]. To evaluate the effect of usage behavior, we introduce a profile described in Section 5.2. Figure 8 illustrates how the shape of the distributions over request-differences change. The experiments were conducted in Linux for Loading events by running a 720p video on the browser in a loop.

Browsing privacy. Inactive users of CoverUp potentially reveal their browsing behavior to the CoverUp server, as a malicious CoverUp server can read HTTP header's referrer field. This leakage is inherent in our approach to use an entry server and to utilize passive participants to produce cover traffic. While this leakage exists, we would like to put it into perspective. Many popular websites already leak this information to other services, such as advertisement networks or external analytic tools, such as Google Analytics.

Neglecting the generalization error. We are aware that it would be useful to have a bound on the generalization, but the limited amount of our experi-

ments does not enable us to properly bound the generalization error with standard techniques.

6 Ethical considerations

"Forced" participation has to be carefully implemented to avoid ethical.We address potential ethical considerations that stem from triggering visitors of some webpage into passively participating in an ACN. Our work received formal approval of our Institutional Review Board (IRB).

Are computation and bandwidth resources of passive participants unwittingly utilized? No, only after an informed consent does CoverUp turn an entry server visitor to a passive participant, hence utilizing the computation and bandwidth resources. Additionally, passive allocation of resources is nothing unexpected for a visitor of a webpage; it is already done by advertisements or QoS scripts, such as Google Analytics. Consequently, webpages that incorporate CoverUp would not cause unexpected behavior on a visitor's browser. The computational overhead of CoverUp is negligible and the bandwidth overhead for a visitor would be around 20.25 MB per day (for a throughput of 10 Kbit/s), which is negligible compared to the data load of video streaming services.

Is CoverUp *harmful to passive participants?* No, CoverUp utilizes standard browser functionality.

Does CoverUp *store potentially incriminating data on the machine of passive participants?* No, we carefully incorporated an All-or-Nothing scheme such that passive participants never contain any useful information on their machine, as long as they do not actively extract and collect the CoverUp data packets from the browser's local storage.

Does CoverUp *trigger passive participants to open potentially suspicious connections?* After an informed consent, CoverUp does trigger a connection to the ACN, which some parties (e.g., an employer) could indeed view as suspicious.[7] We do not consider this an ethical issue since this connection is only opened after information about this connection was provided and an active consent was received.

Does the CoverUp *server collect information about the browsing behavior of the entry server's visitors?* No, while each iframe request of every entry server's visitor includes the visitor's IP address, the CoverUp server does not collect or store this information in any form and immediately deletes it.

7 Selected legal questions

One of the challenges in answering the question whether the provision of CoverUp and the upload of the JavaScript code by the entry server is legal or not (and many other questions evolving around the use of the Internet) is that, whereas the Internet functions globally, law mostly [8] remains limited by territory because sovereign states put their own legislation into effect [3, 6, 17]. The legal provisions and possible offenses that apply to the technical setup of CoverUp, differ from country to country. Moreover, as law is not an exact science and definite legal statements are made by the courts, we conclude the legal discussion herein with an assessment that we consider probable.

In this section we limit the legal analysis to a selected discussion on whether the activity of the provider of the entry server could qualify as cybercrime offense. We do not, for instance, analyse offenses by the provider of the CoverUp server or of the ACN, or cover aiding and abetting.

Many countries enforce their own laws and have their own (territorial) jurisdiction, many countries, among others the EU member states and the USA, have ratified [5] in the Convention on Cybercrime [8] (CCC) – the international treaty on crimes committed via the Internet and other computer networks. This international treaty criminalizes, among others, illegal access (Art. 2 CCC), data interference (Art. 4 CCC), and misuse of devices (Art. 6 CCC).

7.1 Passive participants

Illegal access. Illegal access (Art. 2 CCC) penalizes the entering of a computer system but does not include the mere sending of an e-mail message or a file to a system. The application of standard tools provided for in the commonly applied communication protocols and programs is not per se "without right", in particular not if the accessing application can be considered to have been accepted (e.g. acceptance of cookies [12–14, 23] by client). However, a broad interpretation of Art. 2 CCC is not undisputed (refer [8], §44 - 50).

7 We assume that the CU:Feed connects to the Feed server via an ACN.

Upon request, the entry server delivers a webpage that contains an iframe request for the CoverUp server, which then delivers the JavaScript to the browser for the download of the packet. Not only does the entry server merely send a file (pointer) to the browser, but the request to download the JavaScript from the CoverUp server is standard browser functionality for communication. The same would happen if the entry server were financed by online advertising: upon request the entry server would deliver a webpage pointing to the advertising server and trigger the download of the advertising text or pictures to the browser. As this is a standard online process, we conclude that even in a broad interpretation of Art. 2 CCC, the provider of the entry server should not be illegally accessing the browser.

Data interference. Data interference (Art. 4 CCC) penalizes the damaging, deletion, deterioration, alteration, or suppression of computer data "without right". This provision protects a computer device from the input of malicious code, such as viruses and Trojan horses as well as the resulting alteration of data. However, the modification of traffic data for the purpose of facilitating anonymous communications (e.g., the activities of anonymous remailer systems)should in principle be considered legitimate protection of privacy (refer [18, 20–22], [15, Recitals(1) and (35)]), [19, Art. 13], and, therefore, be considered as being undertaken "with right" [8, §61].

CoverUp does not damage, delete, deteriorate, or suppress data on the participant's client. However, it does alter the data on the hard disk: on the one hand the webpage with the iframe uses disk space and thus modifies the participant's data; on the other hand CoverUp triggers the download of the JavaScript code and subsequently the packets from the ACN to the passive participant's browser, which again uses disk space and thus modifies the data anew.

However the explanatory report to the Convention on Cybercrime foresees that the file causing data interference be "malicious". Code is malicious if it executes harmful functions or if the functions are undesirable.

As concluded above, the JavaScript code utililized standard core browser functionality. Thus from a technical viewpoint, CoverUp is not harmful. Therefore in our view the provider of the entry server not does cause any malicious data interference. We advocate that Art. 4 should not apply to the provision of the webpage with the iframe by the provider of the entry server.

Misuse of devices. Misuse of devices (Art. 6 CCC) penalizes the production, making available, or distribution of a code designed or adapted primarily for the purpose of committing a cybercrime offense, or the possession of such a computer program. It refers to the commission of "hacker tools", i.e. programs that are e.g. designed to alter or even destroy data or interfere with the operation of systems, such as virus programs, or programs designed or adapted to gain access to computer systems. The objective element of offense comprises several activities, e.g. distribution of such code (i.e. the active act of forwarding data to others), or making code available (i.e. placing online devices or hyperlinks to such devices for the use by others) [5, §72].

One of the main questions relating to the misuse of devices is how to handle dual use devices (code). Dual use means in our case that the JavaScript code could be used to download legal content, e.g. political information, as well as illegal content, e.g. child pornography. Should Art. 6 CCC only criminalize the distribution or making available of code that is exclusively written to commit offenses or should it include all code, even if produced and distributed legally? Art. 6 CCC restricts the scope to cases where the code is objectively designed primarily for the purpose of committing an offense, therebyusually excluding dual-use devices [5, §72–§73].

First, it is important to note that CoverUp was not designed primarily for the purpose of committing an offense. While the main purpose of CoverUp is to protect privacy, it can be used to conceal illegal activities.

Second, can the download of criminal information be considered an illegal activity if the information is encrypted? Here we draw a legal analogy to data protection law. Data relating to an identified or identifiable person is considered personal data [15, Art. 2(a)], [24, Art. 4(1)]. If a person is identifiable or identified, data protection law applies. However, if the personal data are pseudonymized or anonymized, then data protection law might not apply anymore because the (formerly identifiable or identified) person cannot longer be identified.

Recital (83), Art. 6(4)(e), 32(1)(a) and 34(3)(a) of the new General Data Protection Regulation[8] stipulate that encryption renders the personal data unintelligible and mitigates the risk of infringing the new regulation.

By applying this data protection principle to the encryption of data by CoverUp we can argue that the data provided by the ACN in the packets are not information because the data is unintelligible. Not only does the passive participant not have sufficient data to reassemble the packet to a whole, but the data are en-

8 Regulation (EU), applicable as of 25.5.2018

crypted in such manner that it is impossible to make any sense of it. At least from a theoretical viewpoint the encryption of CoverUp cannot be breached. We therefore conclude that the JavaScript code, with regard to the passive participant, does not qualify as dual use device because even if it is used for illegal purpose. The data transmitted remain unintelligible and therefore do not qualify as information. Moreover, the JavaScript code, with regard to the active participant, can be qualified as dual use device because the encrypted and unintelligible data are decrypted and reassembled to intelligible information.

Legal conclusion. We discussed the applicability of Art. 2 (illegal access), 4 (data interference), and 6 (misuse of device) CCC to CoverUp. We conclude that the provider of the entry server is probably not illegally accessing the participant's browser by applying CoverUp; that the provider of the entry server probably does not cause any malicious data interference; and that the use of CoverUp with regard to the passive participant does not qualify as misuse of device. In regard to the reassembly of the packets to a meaningful whole, if the information is illegal, CoverUp might qualify as dual use device and fall under Art. 6 CCC. We conclude that at least with regard to the risk of indictment pursuant to Art. 6 CCC it seems advisable that the provider of the entry server does not provide the JavaScript code for download.

7.2 Entry servers

A participant is dependent on Internet service providers (ISP). The question arises whether an (ISP) should be liable for illegal Internet activities of its subscribers. In the following we discuss legislation and case law on the ISP's liability in two different jurisdictions: the EU and the USA.For this discussion it is important to differentiate among the various types of ISPs, for instance access providers, hosting providers, and content providers [68].

European union. In the European Union, liability of ISPs has been regulated in the E-Commerce Directive [11]. Generally, providers shall not have any obligation to monitor the information which they transmit or store, or to seek actively facts or circumstances indicating illegal activity [11, Art. 15 (1)]. According to the directive, access providers acting as "mere conduits" shall not be liable for the information transmitted, on the condition that they do not initiate, select the receiver of, or select or modify the information contained in the transmission [11, Art. 12 (1)].[9] Caching providers (efficiency transmitters) shall not be liable for the automatic, intermediate and temporary storage of information, on the condition that they do not modify the information; comply with access regulations and industry standards for updating the information; do not interfere with the lawful use of technology; and act expeditiously to remove information if removed from the initial source [11, Art. 13 (1)]. Hosting providers shall not be liable for the information stored on their servers, on the condition that they are unaware of illegal activity or information or acts expeditiously to remove or disable access to the illegal information [11, Art. 14 (1)].

With regard to the obligations of a hosting provider, the European Court of Justice decided in SABAM v Netlog[10] that, among other directives, the E-Commerce Directive precluded a national court from issuing an injunction against a hosting service provider which requires it to install a system for filtering (a) information which is stored on its servers by its service users, (b) which applies indiscriminately to all of those users; (c) as a preventative measure; (d) exclusively at its expense; and (e) for an unlimited period; which is capable of identifying IP-infringing content.

USA. Similarly, in the United States there are limitations on liability relating to material online [1]. There are statutory limitations for transitory communications (i.e. access provider, "mere conduit") [1, Section 512(a)], system caching (i.e. storage for limited time) [1, Section 512(b)], information residing on systems or networks at the direction of users (i.e. hosting) [1, Section 512(c)], and information location tools (i.e. search engines or hyperlinking) [1, Section 512(d)].

With regard to the obligations of a hosting provider [1, Section 512(c)], the United States Court of Appeals for the Second Circuit, by referencing UMG Recordings, Inc. v. Shelter Capital Partners LLC, 667 F.3d 1022 (9th Cir. 2011), argued that "*[t]he Court of Appeals affirmed [...] that the website operator was entitled to safe harbor protection. With respect to the actual knowledge provision, the panel declined to 'adopt [...] a broad conception of the knowledge requirement,' id. at 1038, holding instead that the safe harbor '[r]equir[es] specific knowledge of particular infringing activity,' id. at 1037. The Court of Appeals reach[ed] the same con-*

9 With regard to the German liability for interference ("Störerhaftung") according to Sommer unseres Lebens (I ZR 121/08), see also decision by the ECJ in Mc Fadden (C- 484/14).

10 ECJ C-360/10.

clusion' [..] noting that [w]e do not place the burden of determining whether [materials] are actually illegal on a service provider.' Id. At 1038 (alterations in original) (quoting Perfect 10, Inc. v. CCBill LLC, 488 F.3d 1102, 1114 (9th Cir. 2007))". Hence, the 2nd Circuit Court concluded, among others, that 17 U.S.C. §512(c)(1)(A) requires knowledge or awareness of facts or circumstances that indicate specific and identifiable instances of infringement.

Legal conclusion. The entry server is probably not an access provider, maybe a caching provider and presumably a hosting provider. In the latter case two points seem relevant: (i) by whom the information is stored on the entry server and (ii) the entry server's knowledge of any (illegal) activity.

First, depending on how the entry server's webpage is set up, the JavaScript code may be stored by the entry server itself or by a third party. Only in the latter case does the provider's liability privilege apply, because if the JavaScript code is stored on the entry server by the entry server itself, then it is neither an access, nor a caching nor a hosting provider, but probably a content provider (assuming that the JavaScript code is qualified as content). The ISP liability privilege does not apply to content providers.

Second, if the JavaScript code is stored by the entry server itself on the entry server, then the entry server obviously has knowledge of the content. The ISP liability privilege should not apply. If the JavaScript code is uploaded by a third party (as done in CoverUp) to the entry server, and the entry server therefore has no knowledge about the content, then under EU and US legislation and case law the entry server should not be held liable for the JavaScript code.

8 Deployment

We have witnessed a steady rise of concern regarding privacy. Such includes state backed surveillance, web based services collecting huge amount of private information and discrimination of citizens who access sensitive materials such as leaked documents. In recent years, a number of countries reformed their privacy protection laws, which specifically aims to provide protections against the misuse of citizens' private data. One major example is European Union's EU-GDPR and the surveys accompanying it [4, 27] shows that there is a need for privacy-preserving systems. Anonymous communication networks (ACN) is the basic building blocks for many privacy preserving protocols. CoverUp provides a strong privacy guarantee for hiding the intention. Our proposed forced participation technique achieves this by hiding the active users in the traffic generated by the inactive users. Existing systems can easily incorporate CoverUp by setting up the entry server in their own service. The code integrating is effortless and requires almost no modification. The host servers only have to include an iframe pointing to the CoverUp server.

9 Proving participation deniability

This section defines the privacy notion that CoverUp achieves: no attacker can tell whether a participant in a protocol is active or passive (Section 9.1).

9.1 Defining participation deniability

As a worst-case assumption, the attacker has complete knowledge about the running time distributions of the active and the passive participants. The attacker is able to control the content sent by the entry server (i.e., the entry server is untrusted) and has full control over the network link (on the Internet Protocol level). In particular, the attacker is able to request arbitrary data and execute arbitrary JavaScript code in the web browsers context of the entry server and is able to drop, modify, and resend any messages sent over the network.

We over-approximate potential previous knowledge by granting the attacker the capability to control the participant's behavior. In particular, this over-approximation avoids the need to deal with various user behavior profiles, since the attacker can choose the user profiles that maximize the leakage.

Of course, a active participant needs a more extended set of input commands than an passive participant. To avert any leakage to the attacker, the commands for both kinds are like for the active one, while for the passive the surplus is simply ignored. Because this attacker might stress the OS infinitely, we restrict the input rate of these commands to a fixed rate t_{user}. Timing leakage is crucial for the notion that we consider. Since interactive Turing machine, or other computation and network models, such as the UC framework, do properly capture timing leakage, we use TUC [32] (a refinement of UC) as network and computation model (see Appendix A for a brief description). TUC enables a normal execution of all protocol parties and an adver-

sary that can measure timing leakage. As the accuracy of a TUC adversary is not limited per se and in many deployed operating systems the accuracy of the timestamps is limited to around 1 microsecond[11], we introduce a limit t_{net} on the sample rate of the attacker.

The challenger Similar to other cryptographic definitions, we use a machine, called the *challenger* Ch, to capture the capabilities and the restriction to the attacker and to define the task that has to be solved by the attacker. This challenger chooses one out of two protocols at random and runs it, one π_I modelling involuntary participant or and the other π_V modelling voluntary participant. The challenger is located between the attacker and the participant, handles all their communication, enforces all restrictions (input rate t_{user} and sampling rate t_{net}). The attacker can intercept any network traffic and controls the entry server. The attacker now has to guess, which scenario the challenger runs. We let the attacker send commands that specify the participant's interaction with the system. As we quantify over all probabilistic poly-time bounded (ppt) machines, we implicitly assume that the attacker has full knowledge about π_I and π_V.

Algorithm 1 describes the challenger $Ch(\pi, t_{net}, t_{user})$. We consider a message in form of a pair (m, p) where m is the message itself in bitstring format and p denotes to the port where m arrives. In our definition there can be three types of ports:

1. u : Denotes to the user port where there can be incoming and outgoing data flow from the browser due to user activities. We denotes the activities as Command (A) and Command (B) which relates to mouse click events on the website of the entry server and the CoverUp/mix server (specified in the command), respectively.
2. N : Denotes to the network port where network leakage in terms of traffic pattern is observed.
3. CS : This the port where the entry server sends and receives data. The outgoing data can be the programmed java script codes and HTML pages. The incoming data consists of the response from the java script code and the HTML page.

The challenger Ch_b relies on two *FIFO* queues Q_{NET} and $Q_{Browser_\pi}$ for input which are populated by network traffic and the browser B_b respectively. Ch_b polls to both of these queues in a predefined time interval t_q.

Algorithm 1: Challenger $Ch(\pi_b, t_{net}, t_{user})$

Notation: Ch challenger, $\mathcal{A}$ adversary, π_b protocol $(b \in \{I, V\})$, $p \in \{\mathsf{network}, \mathsf{user}\}$ the interface over which the message comes

Upon Initialization
begin
  Initialize two empty FIFO queues Q_{net}, Q_{user}

Upon Receiving m from the $\mathcal{A}$ over interface p
begin
  if $p = \mathsf{user}$ **then**
    $Q_{user}.\mathsf{push}(m)$
  else if $p = \mathsf{network}$ **then**
    $Q_{net}.\mathsf{push}(m)$

Invoke every t_{net} point in time **begin**
  $(m_1, m_2) \leftarrow Q_{net}.\mathsf{pop}()$
  Send (m_1, m_2) over the network port to π_b

Invoke every t_{user} point in time **begin**
  $(m_1, m_2) \leftarrow Q_{user}.\mathsf{pop}()$
  Send (m_1, m_2) as user inputs to π_b

Whenever π_b outputs m over the interface p **begin**
  Send (m, p) to $\mathcal{A}$

Example 1: Instantiating the model with CoverUp. In the case of CoverUp, the scenario π_I constitutes a Chrome browser together with our external application, which visits an entry server. The attacker determines what the participant does on the entry server. Only the uni-directional without any external application is used. In the other scenario π_V, the CoverUp extension is installed in the Chrome browser, together with our running external application. Here, the user utilizes the external application explicitly. The attacker tries now to distinguish to which scenario applies to a specific participant. To accomplish that, he gives commands to the participant as it would be in π_V. If it is in π_I, the additional commands just get ignored. Appendix 9.2 gives a full description of π_I and π_V. ◇

Along the lines of other indistinguishability-based definitions, we compare the probabilities of two interactions: a ppt machine $\mathcal{A}$ (the attacker) either interacts with the challenger that internally runs (i) π_I or (ii) π_V. We require that no ppt attacker $\mathcal{A}$ can distinguish case (i) from case (ii). Technically, no ppt machine $\mathcal{A}$ shall have a higher probability to output (as a final guess) 0 in case (i) has more than a distance δ away from the

11 Often the timestamp also contains nanoseconds but the accuracy is nevertheless in microseconds.

probability that $\mathcal{A}$ outputs (as a final guess) 0 in case (ii). In contrast to other indistinguishability-based definition and similar to differential privacy [49], we do not require δ to be negligible in the security parameter, as we also want to capture systems that do have a small amount of leakage, such as CoverUp, which is however even after thousands and even millions of attacker-observations still small. Throughout the paper, we use the notion of an *attacker's accuracy*, i.e., his probability to guess correctly. A notion that is also used in the context of classifiers. The δ from the defintion below can be converted into the typical notion of an attacker's advantage by $advantage = (\delta - 0.5) \cdot 2$.[12]

Definition 2. *A pair of protocols (π_I, π_V) has* δ-participation deniability *if and only if there is a δ such that for all probabilistic poly-time machine $\mathcal{A}$ we have*

$$\Big|\Pr[0 \leftarrow \langle \mathcal{A} \mid Ch(\pi_I, t_{user}, t_{net})\rangle)] - \Pr[0 \leftarrow \langle \mathcal{A} \mid Ch(\pi_V, t_{user}, t_{net})\rangle)]\Big| \leq (\delta - 0.5) \cdot 2$$

where $b \leftarrow \langle X \mid Y \rangle$ denotes the interaction between the interactive machines X and Y with output b. This interaction stops whenever X stops, and the output b of the interaction is the output of X after it stopped.

Recall that the notion of differential privacy additionally includes a multiplicative factor ε. While our definitions and results could be generalized to such a multiplicative factor ε – ending up with computational differential privacy –, we omitted the ε (thus concentrating on $\varepsilon = 0$) in order to simplify the interpretation of our definition and results.

Related formalisms. The total variation (sometimes called statistical distance) δ that we require is equivalent to $(\epsilon, \delta)-$Differential Privacy with $\epsilon = 0$. Moreover, the total variation accuracy is connected to Kullbeck-Leibler divergence by Pinsker's inequality, and hence to relative entropy.

9.2 Description of protocols

We use two protocols namely π_I and π_V to denotes the sequence of executions performed by the inactive and active participants respectively. A protocol π is defined as a quadruplet $\{B, \Pi, S_{\text{CoverUp}}, S_{mix}\}$. Where $B, \Pi, S_{\text{CoverUp}}$ and S_{mix} denotes a browser, a set of operations, CoverUp server and the mix server respectively. A browser B is defined as an interactive Turing machine. B has three ports namely user port, network port and output port. B takes input from a user (and/or an attacker $\mathcal{A}$) via the user port. All requests and responses to and from remote servers such as S_{CoverUp} and S_{mix} is done via the network port which provides a secure communication channel. The output port is utilized to send any output result to the user e.g. data received from S_{mix}. Additionally a browser can also execute a JavaScipt code which in turn can send request over network to a specific recipient (e.g. S_{mix}) via the network port. Upon receiving response, browser can send it to the user (and/or the attacker). We specify two instance of browser, namely B_I and B_V corresponding to π_I and π_V respectively. The browser instance B_V is B_I with an extension E installed in it. E is also an interactive Turing machine which introduces additional transitions to B. Hence $B_V = B_I \| E$.[13]

9.3 Reduction to timing leakage

Games 1-3 describe hybrid games which incorporate small changes over Game 0 or $Ch(\pi_I)$ (protocol executed by passive participants) and transform the protocol to $Ch(\pi_V)$ (protocol executed by active participants) by adding up a small amount of delay.

Lemma 2. *Assume that π_I and π_V established a secure channel in TUC (i.e., TLS in the real implementation).*[14] *Assume that all cryptographic primitives in π_V and π_I and secure in the TUC framework. Let $\pi_I + \Gamma$ be some protocol that behaves just like π_I, except that it incurs additional delays, which add up to Γ. Then, there is a Γ such that $\pi_I + \Gamma$ and π_V are indistinguishable in the sense of Definition 2 with a $\delta = \mu$ for a function μ that is negligible in the security parameter. Moreover,*

12 For a set of true positives TP, false negatives FN, true negatives TN, and false positives FN, Definition 2 can be rewritten as $|TP|/|TP \cup FN| - |FP|/|FP \cup TN| \leq \delta$. For $accuracy = |TP \cup TN|/|TP \cup FN \cup FP \cup TN|$, if $|TP \cup FN| = |TN \cup FP|$ then we get $2 * accuracy - 1 \leq \delta$.

13 $\|$ is defined as the combination of two state machine as described in https://www.cs.cmu.edu/afs/cs/academic/class/15671-f95/www/handouts/shared-memory/node1.html

14 Formally, a functionality $\mathcal{F}_{SCS}$, as in the UC framework, but secure in TUC (see Appendix A).

Protocol 1 B_I : abstraction of the browser in π_I (feed)

Upon Connecting to the entry server and receiving an iframe
begin
  Compose request r from the iframe
  Send r to CoverUp server via secure channel
Upon Receiving a JavaScript code `code` from CoverUp server
begin
  Execute `code` **begin**
    $h_f \leftarrow$ feed
    $data \leftarrow \{0\}^k$
    send $(h_f \| Data)$ via the secure channel to the mix server
Upon Receiving D from mix server over the secure channel
begin
  Send D to the user

Protocol 2 B_V : abstraction of the browser in π_V (bi-directional channel)

Upon Connecting to the entry server and receiving an iframe
begin
  Compose request r from the iframe
  Send r to CoverUp server via secure channel
Upon Receiving a JavaScript code `code` from CoverUp server
begin
  Execute `code` **begin**
    $Data \leftarrow$ `readBiDirectionalData()`
    Set $h_f \leftarrow$ bi − directional
    Set $ID_{int} \leftarrow \{0,1\}^k$
    Send $(h_f \| ID_{int} \| Data)$ to the mix server
Upon Receiving D from mix server over the secure channel
begin
  Send D to the user

Protocol 3 CoverUp (r): CoverUp server side computation

Upon Receiving a request r from a browser $B_{I/V}$
begin
  `code` ← JavaScript code snippet
  Send `code` to $B_{I/V}$

Protocol 4 mix server(h_f) : the mix server side constant time computation

Upon Receiving $(h_f \| ID_{int} \| Data)$ from the secure channel
begin
  $FixedExecutionTime \leftarrow x$
  $start \leftarrow$ `timeNow()`
  if $h_f = bi - directional$ **then**
    Initialize $state$ with ID_{int}
  if `stateExists()` = TRUE **then**
    Set $state \leftarrow$ `getState`(ID_{int})
    $D \leftarrow$ `covertData`$(state, Data)$
    Call `UpdateState`$(state)$
  else
    $D \leftarrow$ `broadcast`
  Sleep for $(x - (\texttt{timeNow}() - \texttt{start}))$
  Send D over the secure channel

Game 1

Ch: **Upon** Receiving (m, p) from $\mathcal{A}$ for client
begin
  if $p = u$ & $m = (\mathsf{Command}(m_1), \mathsf{Command}(m_2))$ **then**
    if $b = 0$ **then**
      Send $(\mathsf{Command}(m_1), \mathsf{Command}(m_2))$ over u to $\mathsf{Browser}(\phi)$
    else
      Send $(\mathsf{Command}(m_1), \mathsf{Command}(m_2))$ over u to $\mathsf{Browser}(\phi)$
  else
    Send m over p to $\mathsf{Browser}(\phi)$
π_I^1: **Upon** receiving a bi-directional request R from ω_U
begin
  `RequestQueue.push`(R)

Game 2

π_V^2: **Upon** Initialization at Ch side
begin
  $Data \leftarrow$ `readBiDirectionalData()`
  Set $h_f \leftarrow$ bi − directional
  Set $ID_{int} \leftarrow \{0,1\}^k$
  Send $(h_f \| ID_{int} \| Data)$ to the mix server

the timing leakage of $\pi_I + \Gamma$ and π_V is 0 for any sampling rate.

Game 0 is defined as $Ch(\pi_I, t_{net}, t_{user})$ which is the challenger choosing protocol with only feed capability (no interactive capability supported). Where as $Ch(\pi_V, t_{net}, t_{user})$ is the challenger who picks protocol

Game 3

1 π_V^3: **Upon** Receiving a request $R = (h_f \| ID_{int} \| Data)$ at the mix server side
begin
2 $D \leftarrow$ mix server(h_f) (the Protocol 4 for the mix server)
3 Send D to Ch

instance supporting interactive communication (browsing and chatting). We assume that the operation that performs on `RequestQueue` introduces Δ_0 delay. Reading of bi-directional data from the external application imposes Δ_1 delay and modifying variables in the request payload introduces Δ_2 delay. We define S_i to be the number of operation executed in game i and $\Pr[S_i]$ denotes the probability that an attacker can distinguish game i by observing the total execution time.

Game 1-3 introduce small modification over base protocol, i.e., the broadcast channel. Every game add some small timing delay Δ to the previous game which is the only information available to $\mathcal{A}$. Henceforth we define the following notations

Notation 1.

$$\Pr[S_i + \Delta] := \Pr[0 \leftarrow \langle \mathcal{A} \mid \textit{Game } i \textit{ with delay } \Delta \rangle]$$

Notation 2.

$$\Pr[S_i - \Delta] = \Pr[S_j] :\Longleftrightarrow \Pr[S_i] = \Pr[S_j + \Delta]$$

We have also used the following relation throughout our proof which can be proved easily.

$$\begin{aligned}
&\Pr[S_j] = \Pr[S_i + \Delta] \wedge \Pr[S_i] = \Pr[S_k + \Delta'] \\
&\Longleftrightarrow \Pr[S_j] = \Pr[S_k + \Delta + \Delta'] \\
&\Pr[S_i + \Delta] = \Pr[S_j + \Delta + \Delta'] \\
&\Longleftrightarrow \Pr[S_i] = \Pr[S_j + \Delta'] \quad \text{(trivial using Notation 2)}
\end{aligned}$$

9.3.1 Game 0 and Game 1

Game 1 only include one operation on `RequestQueue` which imposes Δ_0 timing delay. Hence

$$\Pr[S_1] = \Pr[S_0 + \Delta_0].$$

$$\frac{\Pr[S_0]}{\Pr[S_1]} = \frac{\Pr[S_0]}{\Pr[S_0 + \Delta_0]}$$

9.3.2 Game 1 and Game 2

Game 2 adds the request intercept which executes in the browser extension at client side. This includes one call to `readBiDirectionalData()` method which reads bi-directional data sent by the external application. This incurs Δ_1 timing delay. Moreover Game 2 adds statements which modify the payload content such as the header h_f to $\mathsf{bi-directional}$ and data field to the bi-directional request. This introduce Δ_2 timing delay. Remember that ll the communications are done via a secure channel, and all the modification of the packet data ensures constant data size. Hence we can ensure indistinguishability in spite of the data modification.

$$\Pr[S_2] = \Pr[S_1 + \Delta_1 + \Delta_2].$$

$$\frac{\Pr[S_1]}{\Pr[S_2]} = \frac{\Pr[S_1]}{\Pr[S_1 + \Delta_1 + \Delta_2]} = \frac{\Pr[S_1]}{\Pr[S_0 + \Delta_0 + \Delta_1 + \Delta_2]}$$

9.3.3 Game 2 and Game 3

In Game 3 all the statements remain same, only the parameter to the reactive machine mix server(h_f) changes as the Ch now sends $\mathsf{bi-directional}$ as the packet header. As mix server(h_f) guarantees constant time execution irrespective of the input parameter, Game 3 does not introduce any additional timing delay.

$$\Pr[S_3] = \Pr[S_2 + \Delta_1 + \Delta_2].$$

$$\frac{\Pr[S_2]}{\Pr[S_3]} = \frac{\Pr[S_2]}{\Pr[S_2 + \Delta_1 + \Delta_2]} = \frac{\Pr[S_2]}{\Pr[S_0 + \Delta_o + \Delta_1 + \Delta_2]}$$

Proof. Game 3 adds total $\Delta_1 + \Delta_2$ delay (cumulative from Game 1 to 3) to Game 0 or $Ch((\pi_I), noise, T_{user}, T_{net})$. Hence

$$\Pr[S_3] = \Pr[S_0 + \Delta_0 + \Delta_1 + \Delta_2] \qquad \text{(eq 2)}$$

holds from eq 1 □

Lemma 3. *Game 3 is equivalent to the challenger $Ch(\pi_V, T_{user}, T_{net})$ who picks the* CoverUp *instance with covert communication mode.*

$$\Pr[S_3] = \Pr[0 \leftarrow \langle \mathcal{A} \mid Ch(\pi_V, t_{user}, t_{net}) \rangle]$$

Proof. From Lemma 2 we get $\Pr[S_3] = \Pr[S_0 + \Delta_0 + \Delta_1 + \Delta_2]$

$$
\begin{aligned}
S_0 &= \mathsf{Step}(\phi + \pi_V) = \mathsf{Step}(\pi_V) \\
S_3 &= \mathsf{Step}(\pi_I^3) = \mathsf{Step}(\pi_V) + \Delta_0 + \Delta_1 + \Delta_2 \\
\Pr[S_3] &= \Pr[S_0 + \Delta_0 + \Delta_1 + \Delta_2] \qquad \text{(from eq 2)}
\end{aligned}
$$

$$
\begin{aligned}
\Pr[S_3] &= \Pr[0 \leftarrow \langle \mathcal{A} \mid \text{Game } 3 \rangle] \\
&= \Pr[0 \leftarrow \langle \mathcal{A} \mid \text{Game } 0 + \Delta_0 + \Delta_1 + \Delta_2 \rangle] \\
&= \Pr[0 \leftarrow \langle \mathcal{A} \mid \text{Game } 0 + \Delta \rangle] \\
&= \Pr[0 \leftarrow \langle \mathcal{A} \mid Ch(\pi_I, t_{user}, t_{net}) + \Delta \rangle] \\
&= \Pr[0 \leftarrow \langle \mathcal{A} \mid Ch(\pi_I^3, t_{user}, t_{net}) \rangle] \\
&= \Pr[0 \leftarrow \langle \mathcal{A} \mid Ch(\pi_V, t_{user}, t_{net})]
\end{aligned}
$$

□

10 Related work

Extending the anonymity set via JavaScript. There are previous research works on utilizing visitors of a collaborating website to produce anonymizing cover traffic via JavaScript. Conscript [42] and Adleaks [63] describes upload only uni-directional channel from the users to the mix network. In contrast, CoverUp provides a transport private bi-directional channel. The paper mentioned timing leakage based side channel attacks but evaluation details are missing except power consumption. In contrast to CoverUp, Conscript additionally has deployment hurdles, since it trusts the entry server to achieve code integrity. While previous work suggests mitigating this trust assumption by letting the extension check all dynamic content to achieve code integrity against a malicious entry server, such dynamic checks will tremendously increase the timing leakage, and thus rendering the active participants clearly distinguishable from passive ones. The need to trust the entry server gives the entry server more responsibility and requires a careful evaluation of the entry servers. All the users of Adleaks who opens the specific website execute the JS provided by the advertisement banner. The implementation of Adleaks requires a patched version of the browser. This reduces the set of possible browsers and therefore reduces the anonymity set massively. Detailed privacy analysis is not described in the paper including timing leakages. The paper [33] describes how to include unwilling users to cover server to server communication. All transport between the servers (by passive clients) is not encrypted. This means an inspection of the HTTP body reveals intention. Moreover, the paper lags any implementation details. Additionally, previous works lacks a legal aspects discussion of "forced" participation (see Section 7).

Transport privacy. Ungers et al. in the SoK paper [66] provided comparison of different transport-private ACNs and compared their privacy, usability and adaptation properties. Section 2.2.1 provides detailed discussion on the compatibility of CoverUp with different ACNs.

Anonymous communication protocols. There are numerous approaches to hide a user's traffic. Anonymous communication (AC) protocols hide traffic by rerouting and potentially waiting. Low-latency AC protocols, such as Tor [48] or Hornet [40], are vulnerable to traffic correlation attacks. High-latency mix-nets, such as Mixminion [46], which do not require the user client to continuously send messages to leak a user intend to connect to the anonymity network, which might seem suspicious and prevent a user from using the mix-net client. AC protocols that do require the user client to continuously send messages, such as DISSENT [43] or Vuzuvela [67], still require the active participation of the users in the protocol, which can leak the intention. Our solution can be easier deployed and does not require a sophisticated infrastructure.

Ricochet is a related project: an anonymous chat. Based on Tor's hidden service design, Ricochet implements a privacy-preserving instant messenger. As Ricochet is based Tor, it suffers from Tor's weaknesses, such as traffic correlation attacks and website fingerprinting. As our system is a constant-rate communication system, CoverUp does not suffer from these kinds of attacks. Tor and thus Ricochet leak that a user intends to use Tor. CoverUp's indistinguishability of active and passive participants enables users to deny the intention to participate in the system.

Covert channels & steganography. Covert channels hide whether communication took place, and thus achieve full deniability. As covert channels typically use a piggyback approach to transport data, they depend on existing data streams, resulting in a dependency of the piggybacked system for latency and throughput. Steganography is another approach which is hiding messages in unsuspicious looking data [30, 50, 56]. But once detected, the origin, thus the intention, is obvious. The same applies to Mixing [57]. Off-the-record messaging publishes the MAC key after each talk, rendering it vulnerable against real-time monitoring [35].

McPherson et al. proposed CovertCast, a broadcast hidden in normal video streams like YouTube [59].

Che et al. were able to create a deniable communication channel based on different levels of noisy channels [39]. Deploying that system is, however, require a much higher effort by the service provider (e.g., YouTube) and does not provide any interactive communication like CoverUp. Freewave [55] provides a covet channel where the user can modulate his internet traffic signal into acoustic data and transfer it to a remote server via VoIPs such as Skype. Such system has bandwidth limitation and is vulnerable to attacks described in [52]. SWEET [70] describes a covert channel e-mail communication where the user can send the query to the remote server by using any available mail server. Such system suffered from very low bandwidth and high latency, making them practically infeasible for deployment. CloudTransport [37] introduced covert communication which involves publicly accessible cloud servers such as Amazon S3 which acts as the oblivious mix. But services like this does not provide protection against attackers learning intention. Infranet [51] describes a system executing covert communication using image stenography but also suffers from a low bandwidth.

11 Conclusion

We discussed how Passive Participation can improve the privacy of anonymous communication network (ACNs).By adding passive participants to the anonymity set, we achieve not only an increased anonymity set but also a participation deniability: an attacker cannot tell whether an observed communication stream originates from an active or a passive participant. We experimentally evaluated the degree of privacy for our prototype implementation of CoverUp for CU:Messenger and CU:Feed and found that the timing leakage is acceptable even under half a year of continual observation. Even vor a state-level agency half a year of continual observation (on sub-ms-level granularity) of a single party incurs a significant cost. This approach of Passive Participation can help to bootstrap mid- and high-latency ACNs.

An interesting direction for future work would be measuring the changes in an active user's behavior towards the entry server via a user study.

Our evaluation of the timing leakage of CoverUp (Section 5) shows that the browser extension, used for the bi-directional channel, has some timing leakages. After many careful observations and experiments, we conjecture that the timing leakages arises from the browser's internal scheduler. The extension operates through several layers of heavy abstractions which makes it non trivial to develop extensions which are such timing sensitive in nature. We think that a set modifications in the browser code will solve such problem where all the modifications can be implemented on native code rather than high level abstraction (such as the JavaScript based API). Such modification is nontrivial and requires a high amount of engineering effort, making it out of scope for our current research. But such modification can be an immediate followup of this work and would be a major contribution towards privacy preserving browsing applications.

References

[1] "17 u.s. code para. 512."

[2] "2016 q1 demandware shopping index." [Online]. Available: https://www.demandware.com/uploads/resources/2016_Q1_Demandware_Shopping_Index.pdf

[3] "America's founding documents | national archives." [Online]. Available: https://www.archives.gov/founding-docs

[4] "Attitudes on data protection and electronic identity in the european union." [Online]. Available: http://ec.europa.eu/public_opinion/archives/ebs/ebs_359_en.pdf

[5] "Chart of signatures and ratifications of treaty 185." [Online]. Available: http://tinyurl.com/h8ketgj

[6] "Consolidated version of the treaty on the functioning of the european union." [Online]. Available: http://eur-lex.europa.eu/resource.html?uri=cellar:41f89a28-1fc6-4c92-b1c8-03327d1b1ecc.0007.02/DOC_1&format=PDF

[7] "Content security policy (csp) - google chrome." [Online]. Available: https://developer.chrome.com/extensions/contentSecurityPolicy

[8] "Convention on cybercrime, budapest, 23.xi.2001." [Online]. Available: http://www.europarl.europa.eu/meetdocs/2014_2019/documents/libe/dv/7_conv_budapest_/7_conv_budapest_en.pdf

[9] "Coverup: Privacy through "forced" participation in anonymous communication networks." https://sites.google.com/view/coverup/.

[10] "Cross origin xmlhttprequest - google chrome." [Online]. Available: https://developer.chrome.com/extensions/xhr

[11] "Directive 2000/31/EC of the European Parliament and of the Council of 8 June 2000 on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market ('Directive on electronic commerce'), 2000 O.J. L 178."

[12] "Directive 2002/22/ec of the european parliament and of the council." [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32002L0022&from=EN

[13] "Directive 2002/58/ec of the european parliament and of the council." [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:en:PDF

[14] "Directive 2009/136/ec of the european parliament and of the council." [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:en:PDF

[15] "Directive 95/46/ec of the european parliament and of the council." [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31995L0046&from=EN

[16] "E-commerce kpi study: There's (finally) a benchmark for that." [Online]. Available: https://moz.com/blog/ecommerce-kpi-benchmark-study

[17] "Eur lex." [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=OJ%3AC%3A2012%3A326%3ATOC

[18] "European convention on human rights (ehcr)." [Online]. Available: http://www.echr.coe.int/Documents/Convention_ENG.pdf

[19] "Federal constitution of the swiss confederation." [Online]. Available: https://www.admin.ch/opc/en/classified-compilation/19995395/index.html

[20] "Fourth amendment." [Online]. Available: https://www.law.cornell.edu/constitution/fourth_amendment

[21] "Katz v. united states, 389 u.s. 347 (1967)." [Online]. Available: https://supreme.justia.com/cases/federal/us/389/347/case.html

[22] "Olmstead v. united states, 277 u.s. 438 (1928)." [Online]. Available: https://supreme.justia.com/cases/federal/us/277/438/case.html

[23] "Regulation (ec) no 2006/2004 of the european parliament and of the council." [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32004R2006&from=EN

[24] "Regulation (ec) no 2006/679 of the european parliament and of the council." [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=en

[25] "Rfc 7323 - tcp extensions for high performance." [Online]. Available: https://tools.ietf.org/html/rfc7323

[26] "Rfc 918 - post office protocol." [Online]. Available: https://tools.ietf.org/html/rfc918

[27] "State of privacy report 2015." [Online]. Available: https://www.symantec.com/content/en/us/about/presskits/b-state-of-privacy-report-2015.pdf

[28] "tshark-the wireshark network analyzer 2.0.0." [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[29] "Bitmessage [Online]," Accessed in Feb, 2017. [Online]. Available: https://bitmessage.org/wiki/Main_Page

[30] D. Artz, "Digital steganography: hiding data within data," *IEEE Internet computing*, 2001.

[31] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi, "AnoA: A Framework For Analyzing Anonymous Communication Protocols," in *CSF*.

[32] M. Backes, P. Manoharan, and E. Mohammadi, "TUC: Time-sensitive and Modular Analysis of Anonymous Communication," in *CSF 2014*.

[33] M. Bauer, "New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets."

[34] D. J. Bernstein, *Curve25519: New Diffie-Hellman Speed Records*.

[35] J. Bonneau and A. Morrison, "Finite-state security analysis of otr version 2."

[36] V. Boyko, *On the Security Properties of OAEP as an All-or-Nothing Transform*.

[37] C. Brubaker, A. Houmansadr, and V. Shmatikov, "Cloud-transport: Using cloud storage for censorship-resistant networking," in *International Symposium on Privacy Enhancing Technologies Symposium*, 2014.

[38] Y. Cao, S. Li, and E. Wijmans, "(Cross-)Browser Fingerprinting via OS and Hardware Level Features," in *NDSS 2017*.

[39] P. H. Che, M. Bakshi, and S. Jaggi, "Reliable deniable communication: Hiding messages in noise," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*.

[40] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig, "Hornet: high-speed onion routing at the network layer," in *CCS 2015*.

[41] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *S&P 2015*.

[42] H. Corrigan-Gibbs and B. Ford, "Conscript Your Friends into Larger Anonymity Sets with JavaScript," in *WPES 2013*.

[43] Corrigan-Gibbs, Henry and Ford, Bryan, "Dissent: accountable anonymous group messaging," in *ACM CCS 2010*.

[44] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*, 2013.

[45] G. Danezis, "Statistical disclosure attacks."

[46] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *S&P 2003*.

[47] G. Danezis and A. Serjantov, "Statistical disclosure or intersection attacks on anonymity systems." in *Information Hiding*.

[48] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Tech. Rep., 2004.

[49] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *TCC 2006*.

[50] J. J. Eggers, R. Baeuml, and B. Girod, "Communications approach to image steganography," in *Electronic Imaging*.

[51] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. R. Karger, "Infranet: Circumventing web censorship and surveillance." in *USENIX Security Symposium*, 2002.

[52] J. Geddes, M. Schuchard, and N. Hopper, "Cover your acks: Pitfalls of covert channel censorship circumvention," in *PCCS 2013*.

[53] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A scalable and efficient protocol for anonymous communication," Cornell University, Tech. Rep.

[54] C. Head, "Anonycaster: Simple, efficient anonymous group communication," 2012.

[55] A. Houmansadr, T. J. Riedl, N. Borisov, and A. C. Singer, "I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention." in *NDSS*, 2013.

[56] M. P. R. Kamble, M. P. S. Waghamode, M. V. S. Gaikwad, and M. G. B. Hogade, "Steganography techniques: A review," *International Journal of Engineering*, 2013.

[57] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis, "Towards efficient traffic-analysis resistant anonymity networks," in *ACM SIGCOMM Computer Communication Review*, 2013.

[58] D. J. MacKay, "Fountain codes," in *Communications, IEEE Proceedings-*.

[59] R. McPherson, A. Houmansadr, and V. Shmatikov, "Covertcast: Using live streaming to evade internet censorship," *Proceedings on Privacy Enhancing Technologies*, 2016.

[60] S. Meiser and E. Mohammadi, "Ratio buckets: A numeric method for r-fold tight differential privacy," Cryptology ePrint Archive, Report 2017/1034.

[61] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *USENIXSecurity 17*.

[62] R. L. Rivest, *All-or-nothing encryption and the package transform*.

[63] V. Roth, B. Güldenring, E. Rieffel, S. Dietrich, and L. Ries, "A Secure Submission System for Online Whistleblowing Platforms," in *FC 2013*.

[64] A. Shokrollahi, "Raptor codes," *IEEE transactions on information theory*.

[65] J. K. Sundararajan, D. Shah, and M. Médard, "Arq for network coding," in *ISIT 2008*.

[66] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "Sok: Secure messaging," in *2015 IEEE Symposium on Security and Privacy*.

[67] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[68] R. Weber, *E-Commerce und Recht, 2. Auflage*, 2010.

[69] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale."

[70] W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov, "Sweet: Serving the web by exploiting email tunnels," 2012.

A TUC: A time-sensitive model for networks of machines

For quantifying the privacy guarantees of CoverUp, we use the TUC framework as a time-sensitive model for network of machines. TUC constitutes a model for networks of machines that is time-sensitive. In TUC, time is represented as a rational number, and there is a global time, on which the time of each machine depends. Each machine has a local clock that is a function t in the global time. This function represents potential delays or inaccuracies of the local timer. Moreover, TUC assigns to each machine a speed s. Hence, a machine is after c step at the global time c/s and the local timer of that machine shows $t(c/s)$.

The execution of a network of machines in TUC is conducted by a single machine, called the execution, that runs all participating machines as submachines. This execution sequentially activates each machine, counts the steps that each machine performs, and coordinates the timely sending and receiving of messages between the submachines. Due to the sequential activation of machines, it can happen that one machine is already far in the future compared to all other machines. It is shown [32] that all reasonable activation strategies lead to the same results. As a consequence, we ignore that TUC internally uses sequential activation and treat all machines as if they are executed in parallel and run according to their speed.

A party can consist of several parallel machines (e.g., several CPUs) that communicate to each other.

A.0.1 Timeless environment and attacker

As in the UC framework, TUC includes an environment and an adversary. This environment and this adversary can consist of several machines that work in parallel. A natural way of modeling this capability is to represent the environment and the adversary as a set of parallel machine. While such a model is more accurate, we decided for the sake simplicity to over-approximate this strength of the environment and the adversary by allowing both parties to make an arbitrary (but polybounded) amount of computation steps in one timestep.

A.0.2 Internet topology

As in the UC framework, TUC models how two machines directly communicate to each other. The internet can, thus, be represented by a network of intermediary machines that (if honest) relay the message from the sender to the destination. A partially global attacker can, of course, compromise several of these machines. Hence, we can abstract this network of machines by the information which connections between protocol parties leaks the message to the attacker. In addition to the previous model, we additionally need to specify the delay of the network, i.e., how much time the connection between two parties takes.