Traffic Flow Confidentiality in IPsec: Protocol and Implementation

Csaba Kiraly[1], Simone Teofili[2], Giuseppe Bianchi[2],
Renato Lo Cigno[1], Matteo Nardelli[1], Emanuele Delzeri[2]

[1] University of Trento,
{kiraly,locigno,matteo.nardelli}@dit.unitn.it
[2] University of Rome Tor Vergata,
{giuseppe.bianchi,simone.teofili,emanuele.delzeri}
@uniroma2.it

Abstract Traffic Flow Confidentiality (TFC) mechanisms are techniques devised to hide/masquerade the traffic pattern to prevent statistical traffic analysis attacks. Their inclusion in widespread security protocols, in conjunction with the ability for deployers to flexibly control their operation, might boost their adoption and improve privacy of future networks. This paper describes a TFC protocol integrated, as a security protocol, in the IPsec security architecture. A Linux-based implementation has been developed, supporting a variety of per-packet treatments (padding, fragmentation, dummy packet generation, and artificial alteration of the packet forwarding delay), in an easily combinable manner. Experimental results are reported to demonstrate the flexibility and the effectiveness of the TFC implementation.

1 Introduction

Extensive literature work demonstrates that the traffic pattern generated during online communications carries plenty of information, which can be gathered through specially devised "statistical traffic analysis attacks". These attacks operate irrespective of the deployed encryption means, and allow the extraction, from the statistical analysis of the generated packet sizes and of their inter-arrival times, of valuable confidential information such as the employed applications [1], the application layer protocols [2], the physical devices used [3], or the web page accessed [4,5]. To perform these attacks, a signature for the protocol or the web site to be recognized is typically pre-computed as a set of statistical parameters describing packet size and/or packet inter-arrival time distributions. Flow classification can be performed by matching the actual statistics with the pre-stored signatures. Quite interestingly, accurate flow classification may be obtained even by looking only at its very first packets [2,6]. Statistical traffic analysis attacks have been also employed for the purpose of

This work was supported by the EU IST 6th Framework Program project Discreet, IST No. 027679

breaching security (such as for gathering passwords transmitted over encrypted sessions [7,8]), and for performing passive [9] or active [10] attacks to anonymization (mix) networks, aimed at uncovering the identity of the communicating parties.

To duly protect the privacy of the users, "Traffic Flow Confidentiality" (TFC) mechanisms are necessary. These mechanisms are devised to alter or mask the statistical characteristics of the traffic patterns. Specifically, they operate by combining basic per-packet treatments, which include, but are not restricted to, i) pad their size and/or fragment them to achieve pre-determined packet size statistics (for instance equalizing the size of all transmitted packets or achieving a random packet size distribution), ii) add extra forwarding delay to alter the packet inter-departure time distribution, and iii) generate dummy packets to fill gaps between subsequently transmitted "real" packets and thus contribute to alter the traffic pattern statistics.

The contribution of this paper is twofold. First, we propose a TFC approach embedded as a sub-layer (a separate, self-contained, TFC protocol) of IPsec. We believe that the inclusion of TFC mechanisms in existing and widely deployed standards may significantly improve their adoption. Second, our approach is not bound to provide a "specific" traffic masking pattern, but rather aims at providing a flexible platform, endowed with a set of packet treatment primitives upon which the system deployers may easily configure the traffic masking patterns they deem more appropriate for achieving a given privacy/performance trade-off.

To the best of our knowledge, our approach differs from existing literature in this field as:

i) it supports a vast choice of packet level modifications (including arbitrary modification of packet size through padding or packet aggregation, as well as through fragmentation[2], artificial alteration of the packet forwarding delay, and "traffic padding" through the suitable insertion of dummy packet), in order to prevent traffic analysis attacks, and in contrast to other protocols that aim to protect against one or another specific attack with a limited set of modifications;
ii) it is developed as a flexible suite of easily composable tools rather than as a pre-programmed specific traffic masking technique (for instance, the frequently employed traffic CBR-ization, i.e. transforming traffic into continuous bit rate pattern composed of packets with maximum size);
iii) it is designed to improve not only the security of the communication but also to protect the privacy of the communicating entities. The proposed protocol provides protection either against statistical traffic analysis aiming to break into a system [7,8], as well as against attacks that use traffic pattern analysis to violate user privacy [4,5];

[2] Indeed, fragmentation is a technique traditionally neglected as a tool for traffic masking, as most of the approaches proposed in the past are based on traffic padding and/or dummy packet generation. Conversely, we believe that fragmentation is a highly effective tool as i) it has a very low overhead if compared with padding or dummy packet generation, and ii) it may be selectively employed on the packets, such as the very first in the flow [1,5,6], which are found to provide most of the information useful for the classification algorithms.

iv) it is deployed as an additional protocol part of the generic IPsec security architecture (thus reutilizing Security Association and Policy management of the IPsec framework), rather than being integrated in a specific Mix-like solution [11,12,13]; and
v) it is already implemented in Linux as part of the packet transformation framework introduced in the 2.6 kernel.

The rest of this paper is organized as follows. Section 2 briefly reviews related works and relevant standards. Section 3 details the design of the TFC system as an IPsec sub-layer and discusses relevant implementation issues. Section 4 experimentally demonstrates the operation of the TFC protocol and its associated control solutions. Finally, conclusions are drawn in section 5.

2 State of the Art

Traffic Flow Confidentiality was identified as one of the fourteen security services already in the ISO/OSI Security Architecture (ISO 7498-2) [17]. The standard identifies the physical, network and application layers as candidates for implementing TFC functionality. Albeit being identified already in the late '80s, security protocols did not focus on TFC for long time as the use of Statistical Traffic Analysis as a basis for attacks was little investigated. Recently, as Traffic-Analysis-based attacks gained ground, TFC-like features appeared in newer version of several security protocols and related products.

TLS 1.0 and 1.1 [18] allows for a padding length up to 255 bytes to be added. Unlike earlier SSL versions where the use of padding was restricted to the smallest integer multiple of the block cipher's block length, TLS allows for any length up to 255 bytes, as long as it results in a length being an integer multiple of the block length. As RFC 4346 states: "lengths longer than necessary might be desirable to frustrate attacks on a protocol that are based on analysis of the lengths of exchanged messages".

Version 3 of the IPsec Encapsulated Security Payload [14] introduced arbitrary size packet padding. The newly introduced padding is still limited in its use: there is no field containing the length of the padding or the length of the original datagram, therefore, padding can only be added if the encapsulated PDU contains a specification of the length of the datagram. This prevents the use of padding for several very important protocols, one for all: TCP.

Our TFC protocol overcomes the limitations present in both of these protocols by supporting arbitrary packet padding for any type of encapsulated datagram. It also provides support for more advanced tools such as fragmentation, aggregation, timing and dummy packets.

Some of the low-latency anonymous routing networks also try to address the issue of statistical traffic analysis by providing limited TFC-like functionality. Tarzan [13] uses the CBR approach by establishing bidirectional, time-invariant packet streams between nodes of the network. This "brute force" approach is supported by our protocol, even if we think that a CBR covert traffic is too costly in terms of resulting

overhead and as such should only be used in highly critical cases. In fact, we argue that, a more selective (hence lightweight) use of dummy packet insertion may eventually provide a better performance-protection trade-off. Freedom (see [19] section 4.3.3) removed TFC from its last version 2.1. While this might be interpreted as a sort of position statement against TFC, the authors themselves claim that the limited methods they have used in previous versions are necessary, but not sufficient to protect against some attacks, and that they plan to study the problem of traffic analysis in subsequent versions[3]. Tor [12] uses fragmentation and padding to fixed size cells. Designers of Tor choose not to adopt other TFC mechanisms in the currently available version, postponing the problem until there is a proven and convenient design that improves anonymity. We believe that (especially in the absence of an "all-for-one" widely accepted traffic masking solution) a clever strategy is to decouple the TFC functionality from the anonymous routing platform, provide a separate protocol specifically devised to provide TFC, and use this as a hop-level component of anonymous routing networks.

Several papers addressing the issue of traffic analysis attacks point to the use of traffic flow confidentiality measures stronger than simple padding. [20] discusses the use of dummy packets and packet delaying in anonymous routing networks both against external and against internal attackers. In [21] a strong traffic analysis attack for global adversaries is presented against real-time anonymization networks that use only packet delaying. In [22] another attack is shown to work on the current implementation of Tor. These papers point towards the use of more advanced TFC techniques such as delaying, cover traffic and stream independence as a possible solution. [16] presents a website fingerprinting attack, but also discusses padding strategies as a countermeasure against such attacks. Authors simulated padding modifying their non-protected traces, and found packet padding to be an effective defense against their attack.

3 TFC sub-layer design and implementation

To overcome the drawbacks of previous solutions, we have designed TFC as a separate IPsec sub-layer, thus maintaining backward compatibility with traditional IPsec implementations. The TFC sub-layer is implemented through a neat separation between i) the TFC control logic, namely the algorithms devised to transform a traffic pattern into another one, ii) their protocol support, accomplished through the specification of a TFC header, and iii) the set of basic mechanisms employed by this control logic to modify characteristics at the packet level. Basic TFC mechanisms can be conveniently categorized as follows:

a) Packet forming: devised to alter the packet size; they include packet padding, packet fragmentation, and packet aggregation (multiplexing);

[3] Development of Freedom was finished after this version due to the company ceasing operation, and due to the success of the similar Tor system.

b) Dummy packet management: devised to generate and discard dummy packets, in order to alter the traffic pattern;
c) Packet timing: devised to alter the forwarding latency of packets adding extra per-packet delay.

Our TFC implementation is developed inside the Linux Kernel 2.6, and leverages the XFRM framework [15] deployed for integrating the TFC processing in the IP/IPsec networking stack[4]. Being developed as an IPsec sub-layer, the TFC protocol can easily take advantage of all the existing ESP functionalities (confidentiality, data integrity and authentication), as well as Security Association and Security Policy management.

Packet Forming

To support the three packet forming mechanisms, we have designed an IPsec Header Extension, the TFC Header (Fig. 1), for the encapsulation of the datagram. The TFC header is internal to the IPsec ESP payload, and it conveys the necessary information to restore the original packet (padding removal, reassembly, de-multiplexing). A next header code should be reserved in the ESP trailer to indicate to the receiver that the protocol contained in the ESP payload is TFC: in our experimentation we used the value 253, reserved by IANA for experimentation and testing. The next header field in the TFC header identifies the protocol carried in the payload.

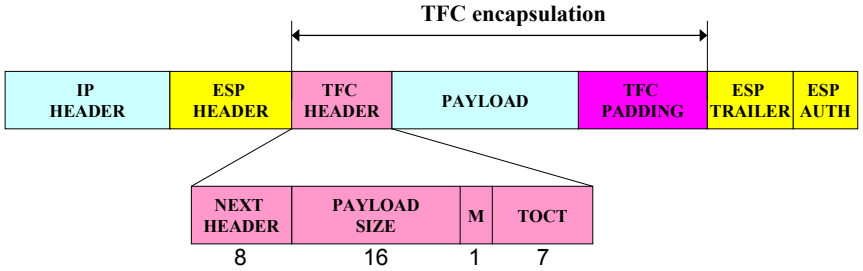

Fig. 1: TFC header format

Packet padding (see Fig. 2) is the traditional (albeit naïve[5]) approach to alter the packet size statistics. The TFC header manages padding simply by explicitly carrying the packet payload size information in a dedicated field. We point out that this allows

[4] In additional details, TFC, just like other IPsec security services, is managed through Security Associations, i.e. we have developed a new Security Association type for TFC similarly as what was done for the ESP and AH SAs. This allows controlling TFC through standard security policies included in the IPsec Security Policy Database (SPD). We use the Netlink interface and XFRM SA and SP databases to implement these features. For additional implementation details, please refer to the Discreet Project Deliverable D3102 – available upon request from the authors.

[5] Indeed, statistics taken on real IP flows show a high variance in the packet size, and thus padding to the maximum possible size introduces a massive overhead. Moreover, studies such as [1,5,6] seem to imply that most of the traffic classification mechanisms use the size information contained in the first few packets of a session which are those that convey the protocol fingerprint, making it less important to `heavily' pad subsequent traffic.

overcoming the significant drawback of the "implicit" padding function proposed in the current version of IPsec, which impedes its usage for inner protocols which do not provide an explicit indication of the payload size (e.g. TCP).

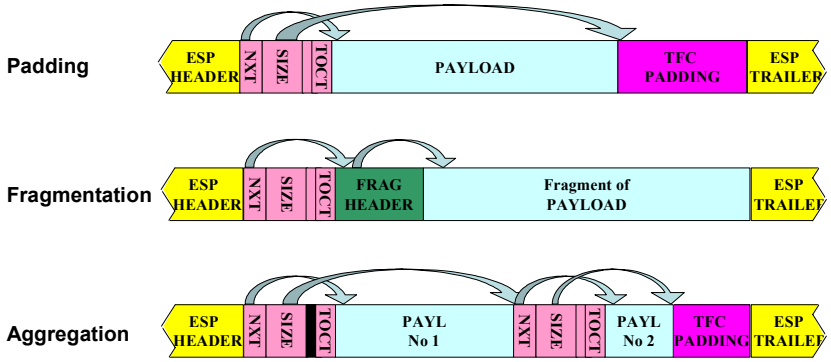

Fig. 2: Padding (using the payload size field); Fragmentation (using fragment header); Aggregation (using the M flag)

Packet fragmentation allows splitting a large packet into smaller packets, and hence avoids the need to add a very large amount of per-packet overhead in the presence of many small packets and a few large ones. The fragmentation feature of the IP header (see Fig. 1) cannot be used for our scope, since it is external to the ESP encryption and thus visible to observers. Therefore, fragmentation should be handled explicitly by our protocol. Instead of placing our own fragmentation related fields in the TFC header structure, we chose to support fragmentation by using an extension header, reusing the structure of the IPv6 fragment header[6] (see Fig. 2). Fragments are reassembled at the end of the overlay link, before the packet is handed out to upper protocols.

Packet aggregation allows multiplexing packets into a bigger datagram, thus increasing the size of the packet through useful information and not through wasted padding bytes. Packet multiplexing is supported by introducing a flag in the TFC header. If this flag is set, after the defined length of the payload, another TFC header and payload follows instead of padding. With this mechanism, several payloads (even fragments) can be transferred in one datagram.

Finally, the TFC header is also exploited to deliver a field, called TOCT (Type of Confidentiality Treatment) which enables to carry information about the type of treatment the packet may be subjected to, when multiple IPsec links are used in a multi-hop fashion, and especially for building IPsec-based mix networks. For reasons of space this operation is not described in this paper.

Dummy Packets

[6] With this choice, we reduce header size when fragmentation is not in use. Moreover, we facilitate implementation of the TFC protocol by exploiting existing implementations of the fragment header and fragmentation functionality in current networking stacks.

The use of dummy (artificially generated) packets is frequently referred to as “traffic padding”. It allows filling traffic gaps and avoids disclosing inactivity periods (i.e., provide unobservability). Moreover, dummy packets are a powerful instrument to alter the traffic pattern statistics, especially when real packets, due to quality of service constraints, cannot be delayed to an extent that allows proper reshaping of the traffic profile. Finally, dummy packet generation is a technique extensively studied for (and employed in some implementations of) anonymization networks to counteract correlation and several types of active attacks (see e.g. [13]).

Protocol support for dummy packet management is straightforward, setting the next header code to "dummy". In our implementation the value 59 is employed, as this value has been standardized in the IPsec ESP specification. For a more homogeneous implementation of the TFC tools we use the dummy packet value 59 inside the inner TFC header, rather than inside the ESP trailer.

Packet Timing

Information extracted from packet inter-arrival times is a usual source for statistical traffic analysis methods [1,5,10]. To counter these attacks, scheduling algorithms (externally programmed in the “control logic” module discussed next – see Fig. 3) should alter the forwarding time of packets.

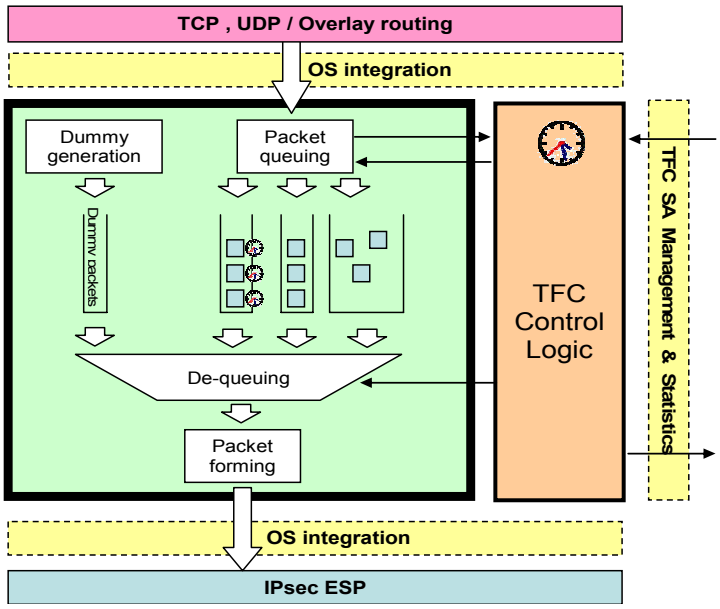

Fig. 3: TFC module architecture. Management and Statistics interface (not detailed it the article) reutilizes management components of the generic IPsec framework

Our implementation supports two methods to alter the packet delays. A first “event-driven” method allows a packet, upon its arrival (top arrow in Fig. 3), to be associated with a specific, possibly packet-dependent, delay. The packet is then

delivered inside the TFC module queues[7] only when the associated timer elapses. A second “timer-driven” method allows de-queuing packets stored inside the TFC module queues at scheduling instants computed according to an algorithm controlled by the Control Logic module. Note that in the case queues are empty, a dummy packet will be de-queued from the dummy packet buffer and delivered.

Implementation complexity is delegated to the control logic implementation, as packet delivery is internally accomplished through appropriate setting of standard Linux timers which drive the invocation of a de-queuing primitive. Trivial methods, such as fixed or random packet clocking, may be easily replaced by adaptive clocking algorithms which explicitly take into account the status of the queues and the related congestion level (although, to the date of writing, the effectiveness of such adaptive approaches in terms of performance/privacy gains and trade-offs is still to be assessed).

Control Logic

The "intelligence" of the system is implemented in a separate control logic module, which can combine the TFC basic mechanisms arbitrarily. For the time being, in order to provide flexibility, we have implemented batching, CBR (Continuous Bit rate), random padding, and random delay algorithms. The ease of such implementation shows the flexibility of the proposed framework, as well as its amenability to implement new algorithms.

We believe that a significant asset of our work is the accomplished decoupling between the algorithmic logic devised to masquerade/shape the traffic pattern, and its underlying implementation as a set of basic tools. This decoupling allows the network deployer to configure (or eventually directly program in the control logic) the most appropriate TFC algorithm suited for its purposes. For instance, in an anonymization network such as Tarzan [13] (which may be built on top of IPsec tunnels extended with the proposed TFC functionalities) the user activity is “well known” but it is necessary to modify the flow fingerprint in order to avoid correlation attack. Conversely, in a point to point connection (e.g., user to proxy) the main goal is to avoid protocol or web site fingerprinting, which may be more adequately countered with different masking algorithms. This versatility of the TFC mechanism also allows for the selection of different protection levels based on user preferences. A simple protection based on packet size modifications can be used for example to achieve a low level of protection. Packet timing can be added to improve the protection in exchange of some performance loss. Finally, the use of dummy packets can be turned on to provide an even higher level of protection, and the control logic can still control the amount of extra packets, from e.g. introducing extra dummy packets randomly amounting to the 10% of original data packets (a strategy that causes only negligible network load increase) to a CBR strategy (which provides perfect protection at the price of a constant load irrespective of the original traffic).

[7] Multiple queues may be internally deployed to differentiate packets incoming from different streams, where a stream may be defined either as a classification rule on the incoming packets, and/or in dependence of the different TFC Security Association mapped over the same IPsec ESP Security Association.

4 Demonstration

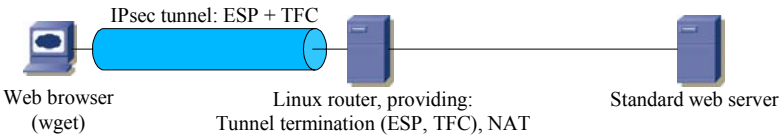

Fig. 4: Test setup

In what follows we demonstrate the flexibility of the TFC protocol as a protection means against website fingerprinting attacks. We chose website fingerprinting as an example where different levels of protection can easily be shown; however we emphasize that this is only one of the situations where Traffic Flow Confidentiality, as a basic security service, has importance. As we have discussed previously, timing based attacks on security protocols, traffic classification attacks, as well as traffic analysis attacks on mix networks all call for a TFC security service, and with different configurations of our TFC protocol (i.e., different control logics configured through SAs and paired with the appropriate SPs), security/privacy of these systems can also be enhanced.

To demonstrate the TFC protocol, we set up a test environment (Fig. 4) similar to the one used in [5] and [16]. We analyze how the information content of traffic dumps can be reduced, while performance degradation remains limited.

In our scenario, a client downloads web pages from normal, unmodified web servers (we use http://www.cnn.com/, http://www.nytimes.com/ and http://www.reuters.com/ for our experiments). To protect against traffic classification attacks, the client creates an IPsec protected tunnel to an exit node. Two ESP SAs and two TFC SAs are set up to cover the traffic of the bi-directional communication[8].

In this setup, protection is provided against attackers eavesdropping on links between the client and the exit node, while traffic is not protected from the exit node or on links between the exit node and the server. We emphasize that this is only an example of the many uses of TFC Security Associations. Being an IPsec security protocol, it can also be used in other network architectures, including security gateway to security gateway tunnels, end-to-end connections (using TFC in transport mode), or cascaded or encapsulated tunnels in anonymizer networks.

The client downloads a full web page, with all of its inline content, using the Firefox browser. We record traces of the packets traveling on the tunnel, repeating the experiment 10 times. We do not consider crypto analysis, therefore, according to our model, the only useful information for the attacker are: the packet length, packet time (and packet inter-arrival time) and the packet direction.

[8] Throughout our tests, we were using symmetric configuration for the SAs, but of course parameters (as well as the control logic) used in the two directions may differ.

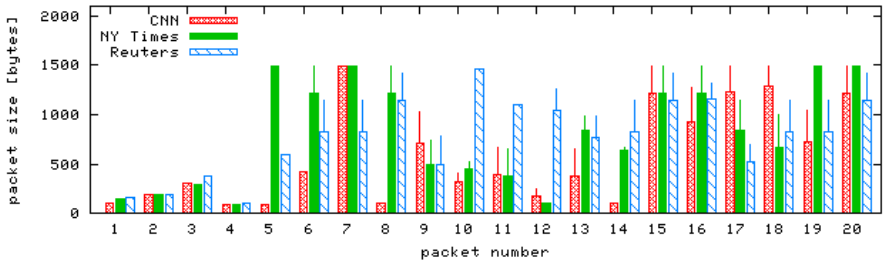

Fig. 5: packet size fingerprints (mean and standard deviation) for three websites.

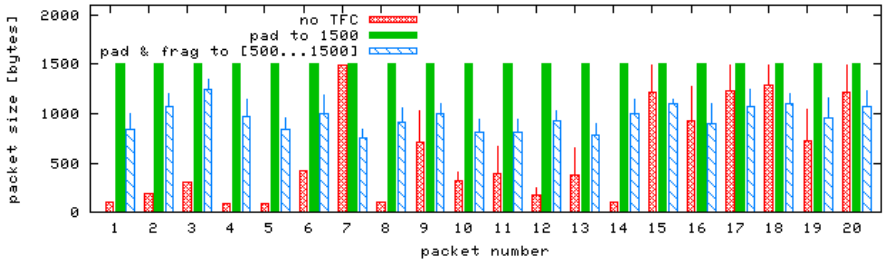

Fig. 6: packet size pattern after TFC is applied

First, we examine protection against attacks based on packet cardinality and properties of individual packets. A usual way of calculating the distance of traffic patterns is to examine properties (size, time or packet inter-arrival time) of the first *n* packets, calculating the distance as a function (typically sum) of individual distances (according to some distance metric) for the 1st, 2nd, …, nth packets.

Fig. 5 shows the size of the first 20 packets for the three sites when no TFC is applied. The filled bars represent the mean packet size, the thin bars the standard deviation. It can be seen how each site has its own characteristic packets (packets where the mean size is different from other sites' mean value, and where standard deviation is low). A traffic analysis tool can recognize these packets and match traffic patterns based on these values.

Fig. 6 shows the size of the first 20 packets after various protections are turned on (we show this only for one website in order to keep the figure readable). This figure demonstrates that the implementation of our protocol is working as expected, but also how characteristic packet sizes can be removed from the trace. Padding all packets to the MTU makes the packet size information uniform over all packets and, what is more important, over all traces. The other control logic determines a random size between 500 and 1500 bytes with uniform distribution, and uses padding and fragmentation to achieve this size. It can be seen (and also follows from the independence of the random value from the original packet size) that the introduced noise is enough to make packet size information useless.

Fig. 7 shows packet inter-arrival times (PIAT) for the three sites. Characteristic values can also be seen in these patterns. In Fig. 8 we show how the introduction of a minimum PIAT, or the introduction of dummy packets with a CBR logic changing the pattern. The first method removes only part of the characteristic values; large PIAT

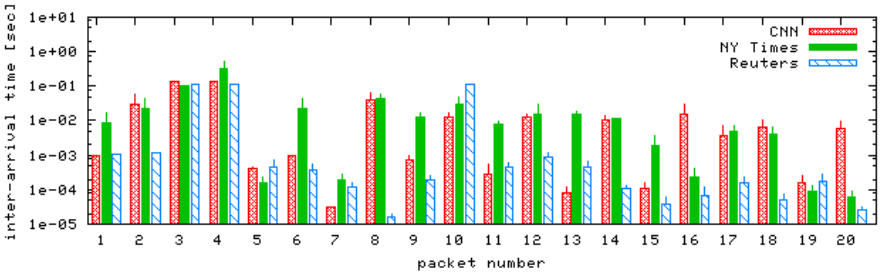

Fig. 7: packet inter-arrival times for the first 20 incoming packets for three websites.

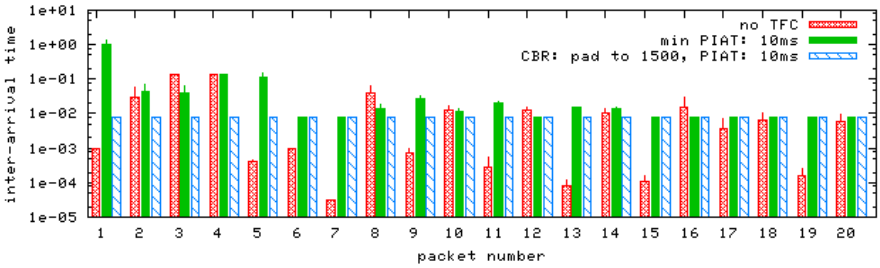

Fig. 8: packet inter-arrival times for CNN after TFC is applied.

values remain almost intact. With the introduction of dummy packets, these can also be changed: CBR with its constant PIAT represents the extreme of the use of dummy packets by removing all information from PIAT values.

How much protection these TFC strategies give? Which control logic to apply? In the figures shown until now, it can easily be seen even visually how information is removed from traffic traces. To really quantify the amount of protection, either an attack should be replicated or an information theoretical measure of the remaining characteristic information should be developed. Such an evaluation is out of the scope of this paper.

However, in Fig. 9 we show a representation, which, according to our knowledge was not used for website fingerprinting attacks before, and provides some insight into the level of protection. While packet counting and packet number based algorithms (such as [2,5,16]) can be fooled using a simple control logic (without randomization and without dummy traffic), looking at Fig. 9, it can easily be seen that the traffic pattern mapped into the "cumulative length" - "elapsed time" space remains somewhat characteristic of the site.

The top figure shows how the download of the three sites can be differentiated if no TFC is applied. For example, the main page of CNN and the NY Times contains the same amount of data and loads almost in the same time, however, differences in the structure of these sites dictate diverse download curves.

On the one hand, we can see that these curves remain (visually) recognizable even after padding (see middle figure) or after simple modifications to timing (see the bottom figure). It is not trivial how this resemblance can be used in automated recognition, but it shows that these simple control logics might not be enough once more sophisticated attacks are developed.

On the other hand, it comes trivially that CBR control logic would change each of the curves into a straight line, and therefore makes attacks based on these curves useless.

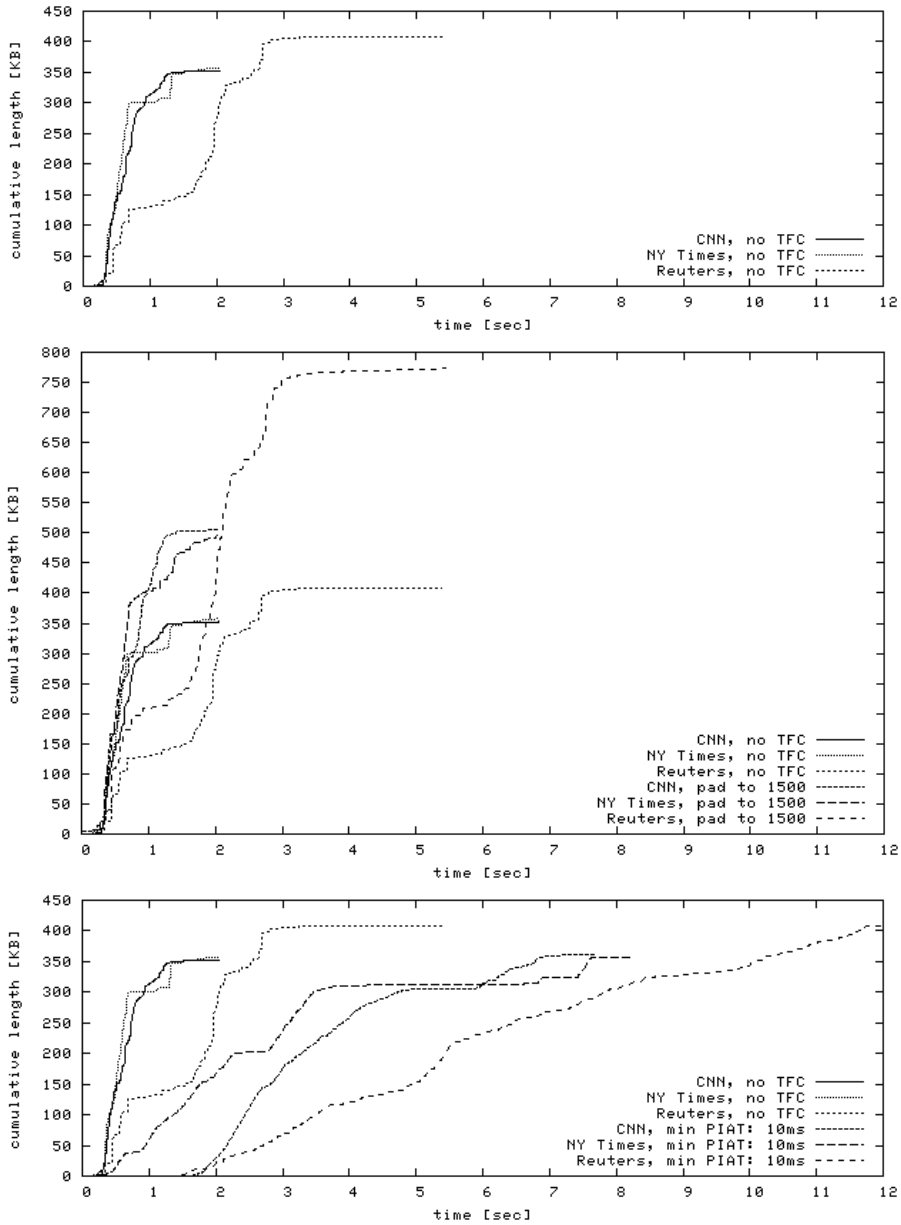

Fig. 9: Download of three web pages with different control logics: (top) representative download curve for each site; (middle) effect of padding on the curves (unmodified curves shown to ease comparison); (bottom) effect of PIAT variation on the curves.

Fig. 9 shows the performance drawback of each control logic as well. The end-point of each line shows the overhead in user perceived page download time and in traffic amount. For example, without TFC, the CNN page downloads in 2 seconds and generates 350 kbytes of traffic. The same download with packets padded to maximum size requires 2 seconds and 500 kbytes (this implicitly means that the tunnel was not the bottleneck). The introduction of a 10ms minimum PIAT limits burst speed to about 140 kbytes/sec, which, with page sizes in these days, reduces user experience considerably (the whole page is downloaded in 7-8 seconds).

These differences justify our choice of separating the control logic from the basic tools. In fact, whenever TFC is required, the appropriate control logic should be selected based on the attack model, performance requirements and the setting in which TFC is applied.

5 Conclusions

As shown by many recent papers, statistical traffic analysis techniques provide good results based on packet size and packet inter-arrival time statistics. We have designed the TFC IPsec security service to protect against such attacks, discussed its implementation and demonstrated its effectiveness.

Our approach goes well beyond current protocols and even beyond protections that were studied against statistical traffic analysis attacks. It provides a variable level of protection that makes the protocol applicable in various systems, introducing fragmentation, aggregation and packet inter-arrival time variation to balance the protection-performance tradeoff.

We further remark that fragmentation, as a technical tool to alter the traffic pattern's statistics, has received little attention in past work which typically implemented traffic flow confidentiality means through packet padding and dummy packet generation. However, literature work on IP flow classification [2] and on web site fingerprinting [6] appear to suggest that a significant amount of the information used by the classifiers is brought by a few packets (frequently the initial ones in the session), whose size is quite different from the remaining ones in the session. This implies that fragmentation (eventually in conjunction with other tools) may be an extremely effective limited-overhead traffic flow confidentiality tool.

Our future work includes the evaluation of different (deterministic and stochastic, traffic independent and adaptive) control logics.

References

1. L. Bernaille, R. Teixeira, and K. Salamatian, “Early Application Identification”, Proceedings of the 2nd ADETTI/ISCTE CoNEXT Conference, Portugal, 2006.
2. M. Crotti, F. Gringoli, P. Pelosato, L. Salgarelli, “A statistical approach to IP-level classification of network traffic”, IEEE ICC 2006, 11-15 Jun. 2006.

3. T. Kohno, A. Broido, K. C. Claffy. “Remote physical device fingerprinting”, in IEEE Symposium on Security and Privacy, pp. 211–225. IEEE Computer Society, 2005.
4. A. Hintz, “Fingerprinting Websites Using Traffic Analysis", Privacy Enhancing Technologies, PET 2002, S. Francisco, USA, April 2002
5. G. D. Bissias, M. Liberatore, D. Jensen, B. N. Levine, “Privacy Vulnerabilities in Encrypted HTTP Streams”, PET 2005, Cavtat, Croatia, May 30-June 1, 2005.
6. L. Bernaille, R. Teixeira, “Early Recognition of Encrypted Application” Proc. PAM, April 2007.
7. D. X. Song, D. Wagner, X. Tian, “Timing analysis of keystrokes and timing attacks on SSH”, 10th USENIX Security Symposium, 2001.
8. B. Canvel, A. Hiltgen, S.Vaudenay, M. Vuagnoux, “Password Interception in a SSL/TLS Channel”, CRYPTO2003, Aug 2003, Santa Barbara, USA
9. Y. Zhu, X. Fu, B. Graham, R. Bettati, W. Zhao “On Flow Correlation Attacks and Countermeasures in Mix Networks”, PET 2004, May 2004
10. X.Wang, S. Chen, S. Jajodia, “Tracking anonymous peer-to-peer VoIP calls on the internet”, ACM Conf. on Computer and Communications Security, November 2005.
11. G. Danezis, R. Dingledine, N. Mathewson, “Mixminion: Design of a Type III Anonymous Remailer Protocol”, 2003 IEEE Symp. on Security and Privacy, May 2003.
12. R. Dingledine N. Mathewson, P. Syverson, “Tor: The Second-Generation Onion Router”, 13th USENIX Security Symp. Aug 2004.
13. M. J. Freedman, R. Morris, “Tarzan: a Peer-to-Peer Anonymizing Network Layer”, ACM Conf. on Computer and Communications Security, Washington, DC, November 2002.
14. S. Kent, "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
15. M. Kanda, K. Miyazawa, H. Esaki, “USAGI IPv6 IPsec development for Linux”, Int. Symp. Applications and the Internet Workshops (SAINT) 2004. pp. 159-163, Jan. 2004.
16. M. Liberatore, B. N. Levine, "Inferring the Source of Encrypted HTTP Connections", CCS2006, October 2006
17. ISO, “Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture”, ISO 7498-2, 1989
18. T. Dierks, E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1”, RFC 4346, April 2006
19. A. Back, I. Goldberg, A. Shostack, “Freedom 2.1 Security Issues and Analysis”, May 2001
20. P. Syverson, G. Tsudik, M. Reed and C. Landwehr, "Towards an Analysis of Onion Routing Security", Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, July 2000
21. G. Danezis, “The trac analysis of continuous-time mixes”, Privacy Enhancing Technologies (PET 2004), May 2004
22. S. J. Murdoch, G. Danezis, “Low-Cost Traffic Analysis of Tor”, In Proceedings of the 2005 IEEE Symposium on Security and Privacy, May 2005.