Network Flow Contracts: Lightweight and Practical Accountability for the Internet

Nikos Ntritsos

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Advisors: Prof. *Evangelos P. Markatos*

University of Crete
Computer Science Department

Network Flow Contracts: Lightweight and Practical Accountability for the Internet

Thesis submitted by
Nikos Ntritsos
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Nikos Ntritsos

Committee approvals: _____
Evangelos P. Markatos, Professor, Thesis Supervisor
Computer Science Department
University of Crete

_____
Apostolos Traganitis, Professor, Committee Member
Computer Science Department
University of Crete

_____
Maria Papadopouli, Associate Professor, Committee Member
Computer Science Department
University of Crete

Departmental approval: _____
Angelos Bilas
Professor, Director of Graduate Studies

Heraklion, April 2014

Abstract

Modern network-forensics tools can trace an attacker based on digital evidence, such as network logs or disk activity. These tools are designed to collect information that can tie criminals to their malicious actions. There are analogous procedures based on evidence, such as DNA or fingerprints, in the physical world. Unfortunately, such evidence can be fabricated for incriminating innocent victims. In the digital world data fabrication is even easier. Network-forensics tools can be intentionally deceived by planting fake digital traces in network logs. The result of such actions can be severe. Innocent users can be accused for taking part in offensive activities - for example, for having visited web sites with pedophiliac content. In this thesis, we introduce Network Flow Contracts (NFCs), a framework that allows users accounting for their network activity by verifying all traces they leave in the last-mile ISP. The framework guarantees that only verified actions can be attributed as authentic. NFCs cryptographically sign every network flow initiating by the requesting user. An ISP providing logs as evidence of a particular user accessing a resource must also provide the cryptographic signature of a network flow initiating from the user to the resource. Generating a fake signature is considered computationally hard, as long as the private key of the user is kept secret. Finally, NFCs are easy to deploy and fast. They introduce zero network lag to the clients and an overhead to the ISP, which can be easily accommodated with todays commodity hardware.

Περίληψη

Τα σύγχρονα διαδικτυακά εγκληματολογικά εργαλεία μπορούν να ανιχνεύσουν ένα επιτιθέμενο βασιζόμενα σε ψηφιακά στοιχεία, όπως είναι τα διαδικτυακά καταγραφόμενα αρχεία ή δραστηριότητα του δίσκου. Αυτά τα εργαλεία είναι σχεδιασμένα να συλλέγουν πληροφορίες, οι οποίες μπορούν να συνδέσουν εγκληματίες με την κακόβουλη δραστηριότητά τους. Στον πραγματικό κόσμο υπάρχουν ανάλογες διαδικασίες βασιζόμενες σε στοιχεία όπως το DNA και το δακτυλικό αποτύπωμα. Δυστυχώς τέτοια στοιχεία μπορούν να αναπαραχθούν και να κατηγορηθούν αθώα θύματα. Στο ψηφιακό κόσμο η παραγωγή στοιχείων είναι ακόμα ευκολότερη. Τα διαδικτυακά εγκληματολογικά εργαλεία μπορούν να εξαπατηθούν σκοπίμως με την προσθήκη ψεύτικων ψηφιακών ιχνών στα διαδικτυακά κείμενα καταγραφής. Το αποτέλεσμα τέτοιων δραστηριοτήτων μπορεί να έχει βαρύνουσα σημασία. Αθώοι χρήστες μπορούν να κατηγορηθούν για συμμετοχή σε προσβλητικές δραστηριότητες, παραδείγματος χάρι επίσκεψη σε ιστοσελίδα με παιδοφιλικό περιεχόμενο. Στο θέσης αυτό παρουσιάζουμε το Network Flow Contracts (NFCs), ένα εργαλείο που επιτρέπει στους χρήστες να ταυτοποιήσουν την διαδικτυακή τους δραστηριότητα στο πρώτο εξυπηρετητή του διαδικτυακού τους παροχέα. Το εργαλείο αυτό εγγυάται ότι μόνο η ταυτοποιημένη δραστηριότητα μπορεί να χαρακτηριστεί ως αυθεντική. Το NFCs κρυπτογραφεί – υπογράφει κάθε διαδικτυακή ροή που ξεκινάει από το αίτημα του χρήστη. Ο παροχέας διαδικτύου πέρα από τα καταγραφόμενα αρχεία στα πλαίσια διατήρησης της κίνησης του χρήστη, πρέπει επίσης να παρέχει την ψηφιακή υπογραφή για κάθε ξεχωριστή ροή που προκαλείτε από τον χρήστη. Η παραγωγή ψεύτικων υπογραφών θεωρείτε υπολογιστικά δύσκολο όσο το ιδιωτικό κλειδί του χρήστη παραμένει μυστικό. Τέλος το NFCs είναι εύκολο και γρήγορο να υλοποιηθεί. Επίσης δεν εισάγει καθυστέρηση τόσο στο δίκτυο του χρήστη όσο στον παροχέα και μπορεί να προσαρμοστεί στους υπάρχοντες εμπορικούς υπολογιστές.

Acknowledgements

I would like to thank my Supervisor, Professor Evangelos P. Markatos, for his valuable guideline and patience during my academic steps in the field of Computer Science.

My sincere thanks also go to Dr. Elias Athanasopoulos for his continuous help and advice in all stages of this thesis.

My best regards to my colleagues in the Distributed Computing Systems Laboratory (ICS/FORTH).

I would also like to thank my parents, my brother and sister. They were always supporting me and encouraging me with their best wishes.

Last but not least, I would like to thank my wife Suela. She was always there cheering me up and stood by me through the good times and bad.

Contents

Chapter 1

Introduction

Digital data can be used for attributing the behavior of users. Special network-forensics tools collect traces and re- veal a user's past activity, which possibly contains illegal actions. During a crime investigation, network traces can be considered as the analogous of human DNA or fingerprints in the physical world. It is challenging, although not entirely impossible, to replicate concrete evidence, such as DNA or fingerprints. On the other hand, it is undemanding to plant network traces in logs and trick network forensics tools. In fact, we believe that fabricating evidence in the virtual world is way easier to do, but unfortunately, has the same consequences as in the physical world. For a sample of cases, where users were wrongly accused based entirely on digital evidence, we refer the interested reader to [12,25,26].

The Data Retention Directive of the European Union, adopted in 2006 [2], requires European ISPs to keep a copy of all Internet accesses their customers have completed for a predefined period of time [2] for facilitating the investigation and prosecution of serious crime, as defined by each Member State in its national law. Similar bills have been introduced, also, in the US [3], and there are even more recent suggestions, followed by many concerns, for data retention in e-mail [23,10]. Although data retention has received a big push by the legislative authorities, today, after six years of the original proposal, data retention achieved only limited harmonisation. This is because there are many concerns about privacy and fabrication of user data. The EU data protection supervisor has included these concerns in a recently published report [5]. More precisely, if data retention is applied, users are obliged to use all Internet services under a big-brother regime. More importantly, it is questionable if all collected information is stored securely at the ISP and if it can avoid tampering or leakage [43]. Users have limited means to protect themselves in the case they are accused based on fabricated data. One can argue that users can take advantage of plausible deniability, but, still, the level of protection they can receive is questionable. Although the authenticity of digital evidence can be denied, since data fabrication is easy, courts reject this argument without proof of tampering [38]. But users, today, have zero technical means to

generate such proofs.

In this thesis we seek for a framework offering practical accountability for Internet users that willing to protect their privacy and to resist any accusations based on fabricated data. We argue that (a) users must be able to receive practical accountability for their network actions from their lastmile ISP, and (b) planting fake information in network logs must be computationally hard.

1.1 Requirements

In this thesis, we seek for a technical framework that can offer protection to an ad hoc innocent user, which is wrongly accused based on digital evidence. The framework's requirements are the following.

1. Protection against data fabrication. Digital evidence can be easily implanted or forged with currently employed technologies. Web servers can be polluted with fake logs, data traces can host foreign records that never existed, and with minimal effort an adversary can create a completely artificial profile for a network user that includes offensive activities. Due to this profile, the victim can be challenged to face a number of negative consequences; from humiliation to conviction for crime [25]. In the general case, during a crime investigation, an ISP is the one who provides evidence about a suspecting network activity. ISPs are considered trusted, since their goal is to provide service to their subscribers, being reliable and competitive with other ISPs. However, modern providers have a large employee base making hard to guarantee that all information stored in the ISP is kept safe from insider threats [41]. In this thesis, we propose a system that requires both the ISP and the user to agree in a contract for every location the user is visiting through the ISP network. The contract, which is essentially a cryptographic signature, encapsulates the user's network activity. In order to reveal the network activity the contract must be revealed. This action requires the contribution from both the user and the ISP. As long as the user is not willing to reveal her private key, the user's network activity cannot be forged, modified or transferred to third parties.

2. Digital alibi. Our system is designed for providing the user with a digital alibi without forcing her to disclose any more information than the minimal required for proving that she has or has not accessed a location near a particular date. Consider that a user has accessed a series of network resources, U . If the user is accused for having visited location u $\in$ U (over a configurable time window), then the user can answer the question without revealing all other accesses, U $-$ u.

3. Preserve of anonymizing systems. Anonymizing systems are hard to run if current proposals for Internet accountability [34, 45] are actually deployed.

Real-world security depends on accountability (imagine, if you will, a world where all actions could be taken anonymously), and we think the same applies to the Internet [8]. We understand that anonymizing systems can be abused for contacting electronic crime. However, anonymizing systems have a huge range of applications that are not related to crime and, most importantly, anonymous free speech is protected by the First Amendment [1]. We seek of a practical method for providing accountability without sacrificing the user's privacy.

4. Practicality and performance. We ideally want simple techniques that do not introduce any computational overhead or network lag in the user's experience. Our framework is based on asymmetric cryptography, which is considered computationally demanding. However, we rarely sign packets (only TCP_SYN) and, most importantly, there is no network lag; clients receive all messages asynchronously. In addition, the cryptographic overhead imposed at the ISP is moderate and can be tolerated with commodity hardware.

5. Incremental deployment. Our framework can be offered by ISPs to users who opt-in for receiving such a protection. There is no need for every ISP implementing the framework, but only by those that are willing to offer protection as an extra service. A user who opts-in for the framework can receive complete protection, without needing any extra cooperation by third parties. In contrast with other clean-state approaches for an accountable Internet, our proposal can be deployed immediately without changing any fundamental concepts.

1.2 Threat Model

We aim at developing a framework for practical accountability of network activity, in order to address a very precise and narrow threat model. Consider Alice, which is ad hoc innocent. Alice is accused of having produced in the past offensive traffic. All accusation has been conducted based on log entries collected at Alice's ISP. The ISP is considered generically trusted. However, we assume that data logs stored at the ISP can be modified via an external intruder or an insider [41]. Alice wants to protect herself against any accusation based on fabricated information stored at the ISP. More precisely, Alice wants to make data fabrication at the ISP computationally hard.

1.3 Proposal

We propose Network Flow Contracts (NFCs), a framework that lets users to record their own Internet logs. The last-mile ISP (i.e., the one a user is using for accessing the network) records Internet access on behalf of its subscribers. To make sure that the ISP does not record more than the user has accessed, users cryptographically

sign every Internet access they make. In a nutshell, the subscriber's host digitally signs every TCP_SYN packet. The ISP, on its part, (i) keeps a record of all signed accesses and (ii) honors all accesses associated with a valid signature. If the ISP receives a request, for which no signature exists, the request does not go through (a TCP-RST packet is sent by the ISP for terminating the connection). Signatures are verified by and stored to the last-mile ISP, only. All other ISPs in the path have not to be aware of anything.

1.4 Contributions

The contributions of this thesis can be summarized as follows.

1. We design and implement a prototype of Network Flow Contracts (NFCs), a framework offering network accountability at the ISP-level. An ISP providing NFCs guarantees that all network activity is logged with the subscriber's verification. All connections that have not been digitally signed are explicitly terminated by the ISP.

2. We evaluate NFCs in a hardware configurations. NFCs impose zero additional network overhead on the clients and moderate overhead at the ISP.

3. We propose a novel storage scheme for keeping all network accesses securely at the ISP. In case the ISP is compromised, the attacker cannot reveal the network history of any subscriber of the ISP. In case an attacker compromises the ISP and the user's private key, it is still hard to reveal all network history of the particular user. Finally, the encrypted logs can be still of use under a crime investigation, if the suspect agrees to provide specific resources encrypted with her private key.

Chapter 2

Click Modular Router Architecture

Click [47] is a new software architecture for building flexible and configurable routers. The Click architecture is centered on the element. Each element is a software component representing a unit of router processing. Individual elements implement simple router functions like packet classification, queueing, scheduling, and interfacing with network devices.

A router configuration is a directed graph with elements at the vertices; packets flow along the edges of the graph. Configurations are written in a declarative language that supports user-defined abstractions. This language is both readable by humans and easily manipulated by tools. Router configurations, in turn, run in the context of some driver, either at user level or in the Linux kernel. Our interest focuses on kernel level.

The Click kernel driver runs as a kernel thread under Linux. The kernel thread loops over the current router's task queue and runs each task. Only interrupts can preempt this thread, but to keep the system responsive, it voluntarily gives up the CPU to Linux from time to time; the user can specify how often with a ScheduleLinux information element.

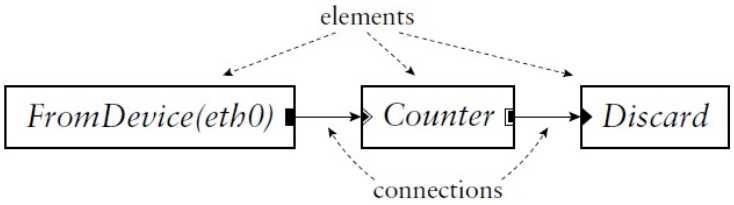


Figure 2.1: A simple Click router configuration.

Figure 2.1 shows some elements [47] connected together into a simple router configuration. Elements appear as boxes; connections appear as arrows connecting

the boxes together.Packets pass from element to element along the arrows (connections). This router's elements read packets from the network (FromDevice(eth0)), count them (Counter), and finally throw them away (Discard).

2.1 Elements

The element is the most important user-visible abstraction in Click. Every property of a router configuration is specified either through the choice of elements or through their arrangement. Device handling, routing table lookups, queueing, counting, and so forth are all implemented by elements. Inside a running router, each element is a C++ object that may maintain private state.

Elements have five important properties: element class, ports, configuration strings, method interfaces, and handlers.

- **Element class.** An element's class specifies that element's data layout and behavior.

- **Ports.** Each element can have any number of input and output ports. Every connection links an output port on one element to an input port on another. Different ports may have different roles. Every port that is provided must be used by at least one connection, or the configuration is in error. Ports may be push, pull, or agnostic.

- **Configuration string.** The optional configuration string contains additional arguments passed to the element at router initialization time.

- **Method interfaces.** Each element exports methods that other elements may access. This set of methods is grouped into method interfaces.

- **Handlers.** Handlers are methods that are exported to the user, rather than to other elements in the router configuration. In the Linux kernel driver, handlers appear as files in the dynamic /proc file system.

2.2 Packets

A Click packet consists of a small packet header and the actual packet data; the packet header points to the data. This structure was borrowed from the Linux kernel's packet abstraction, sk_buff. In the Linux kernel driver, Click packet objects are equivalent to sk_buffs, which avoids translation or indirection overhead when communicating with device drivers or the kernel itself. Operations on an in-kernel Packet have zero overhead over the corresponding Linux sk_buff operations.

Headers contain a number of annotations in addition to a pointer to the packet data. Annotations may be shared with Linux or specific to Click. Some annotations contain information independent of the packet data- for example, the time when the packet arrived. Other annotations cache information about the data. Annotations are stored in the packet header in a fixed static order; there is currently no way to dynamically add a new kind of annotation.

2.3 Connections

A connection passes from an output port on one element to an input port on another. Connections are the main mechanism used for linking elements together; each connection represents a possible path for packet transfer between elements. In a running router, connections are represented as pointers to element objects, and passing a packet along a connection is implemented by a single virtual function call. Connections are drawn as arrows; each arrow's direction represents the direction of packet flow.

Click supports two kinds of connections, push and pull, that implement complementary kinds of packet transfer. On a push connection, packets start at the source element and are passed downstream to the destination element. On a pull connection, in contrast, the destination element initiates packet transfer: it asks the source element to return a packet, or a null pointer if no packet is available.

2.4 Click lacks of elements for asymmetric cryptography

From its original version in 2000 by Eddie Kohler, click is enriched with a large number of elements that enable the user to configure routers that perform complex and sophisticated functions. Although this rapid evolution, in the security sector there was not such a similar progress. Click modular router provides no elements for asymmetric cryptography. Digital signature and the identification of packet's source are important issues that the current click's version does not provide.

The above mentioned functions constitute an important role for the client's protection from criminal and social accusation. The explosion of social network and the daily use of the Internet make the traffic trace an important evidence for the user's daily activity. The identification of the real network traffic traces is solved with the use of asymmetric cryptography. For this reason we extended the click modular router to provide RSA cryptography.

The reason we chose click is because it is a software architecture for building flexible and configurable routers. The creation of complex routers is configured from basic elements, which are independent from each other and performs simple functions. Elements pass packets to one another over links called connections. Each connection represents a possible path for packet transfer. The synthesis of all the connections provides more complex functions. The approach to these small

entities and the convenient connectivity between them makes click easy to manage and create complex routers.

Chapter 3

Architecture

In this section we describe the architecture of the NFCs framework. We begin with a basic high-level overview and then we give definitions and more technical insight about the core components of the framework, i.e. the contracts. We further proceed and discuss a storage scheme, which if employed by an ISP, it can significantly protect the users' data, even if the ISP becomes compromised. Finally, we discuss how our framework guarantees that data fabrication at the ISP level is computationally hard, how the user's privacy is preserved and how NFCs behave under failures.

3.1 Overview

NFCs are offered by ISPs as a service to users willing to protect themselves by authenticating their network activity. NFCs are not a new data retention scheme and do not aim on substituting existing data retention schemes, but a new service for users that care to provably protect their network actions. NFCs must be deployed in the last-mile ISP only; other ISPs along an Internet path do not need to be aware of the scheme. The fact, that subscribers of an NFC-enabled ISP can opt-in or not, and NFCs may be implemented selectively by any ISP willing to offer the service, greatly assist in a realistic and immediate deployment, using the currently available technologies. The basic requirement is that the user runs a custom software for connecting to the Internet, something which is typical. Many ISPs offer their own software for facilitating their users to set-up the Internet connection and the login process. Additional requirement is that the user owns a pair of public and private key, which is also typical; many users use PGP [21] in e-mail, which is based, in part, in public-key cryptography. The final requirement is for the ISP to run a custom software. Modern ISPs run many custom components for managing and accounting all user-generated traffic, so we believe that this is also a typical requirement. In this thesis we present a generic implementation of both the server and client part of the software. In a real deployment, the client part can be implemented at the kernel, by OS developers, or in user space as a standalone

application. In a nutshell, NFCs work as follows:

1. The subscriber registers for the service providing the ISP with her public key.
2. The subscriber installs the software client for signing all outgoing traffic routed by the ISP with her private key.
3. For every new flow created by the subscriber, the ISP waits for time T (we provide typical values of T in Section 5 to receive the flow's signature. If the signature is not received, the ISP terminates the subscriber's connection by explicitly sending a TCP-RST. Termination is done asynchronously and, thus, there is no network lag introduced to the client.

This scheme is applicable to all TCP traffic generated by the subscriber. We consider only TCP traffic, since this type of traffic is associated with state-full network activities.

3.2 Network Flow Contracts

An NFC is a digital contract between a user and the ISP. For all further discussion, we assume a DSL user connected to an ISP, but the scheme is quite generic to be applicable in other cases, such as users connected to a telephony network. For generating an NFC the subscriber is required to own a pair of public and private key, and operate an NFC client. For running the NFC client, the user must provide the software with her private key. The NFC client passively monitors all traffic generated by the subscriber's host. For each outgoing TCP-SYN packet, the client retrieves the destination IP address and port number of the packet, it timestamps and signs them with the user's private key and sends the packet containing the signature1 to an NFC server operated by the ISP. The ISP maintains for each online subscriber two tables: (a) the Connection Table, and (b) the Signature Table.

The ISP generates a record for each incoming TCP-SYN packet received by the subscriber's host. The record is constructed by taking the user's identity and the key produced by hashing the 3-tuple (destination IP, destination port and protocol) of the new TCP connection. We do not rely on the 5-tuple (source IP/port, destination IP/port, protocol), because usually NAT/Proxies can overwrite the source part [15]. The ISP proceeds and routes the TCP-SYN packet to the destination, and in parallel monitors the Signature Table for a signature associated with the new TCP connection. In the case that no signature has been received after T seconds from the time the original TCP-SYN has been observed, the ISP terminates the subscriber's connection by sending an explicit TCP-RST. In Section 4 we estimate appropriate values for T . The NFC protocol is presented in Figure 3.1.

A Signature Table is maintained by an NFC server. The user's NFC client is constantly connected with the ISP's NFC server. This connection communicates

all NFCs (i.e. the signatures) from the subscriber's host to the ISP. For each new outgoing TCP connection, the NFC client sends a special packet to the server. The packet is composed by an identifier, ID, which characterizes the connection and the signature. The ID is formed by hashing the subscriber's name concatenated with the 3-tuple information using SHA1 and the signature is the destination IP/port and the timestamp encrypted with the user's private key. The timestamp is required for protecting the user against replay attacks [42]. We further discuss replay attacks, the transitivity of IP addresses, and why we choose IPs for signatures - and not something more descriptive, like a URL - in Section 5. The packet format is depicted in Figure 3.2 and an overview of the NFC architecture is depicted in Figure 3.3.

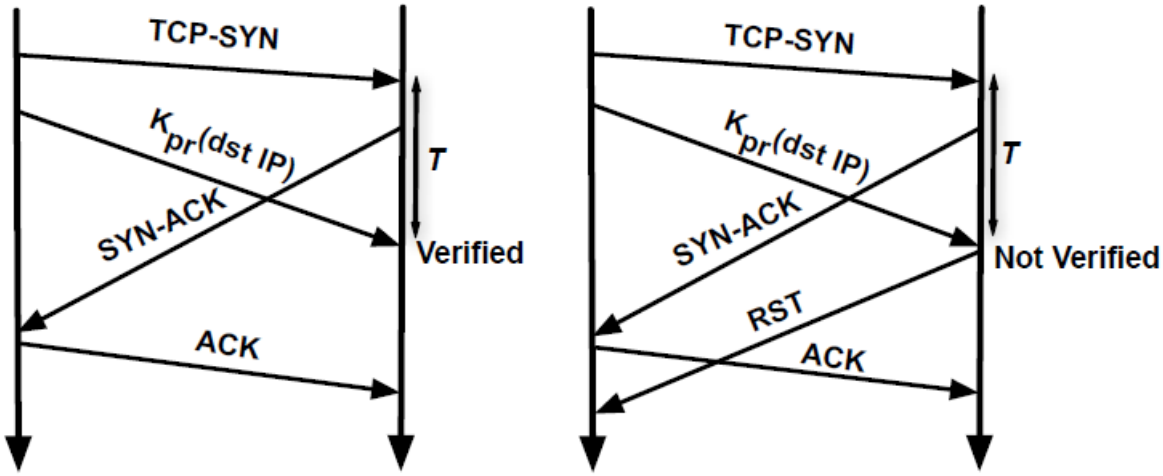


Figure 3.1: The NFC protocol. For every incoming TCP-SYN the ISP waits for a signature. In case no signature is received after T seconds from the time the original TCP-SYN was observed, the ISP terminates the subscriber's connection by sending an explicit TCP-RST.

3.3 Provable Network Activity

NFCs can guarantee that network-forensics tools cannot be deceived by fake information planted in network logs. An attacker who wants to plant false records in network logs has to obtain the private key of the victim. Unless the private key of a subscriber is leaked, it is computationally hard to fabricate the authentic log files.

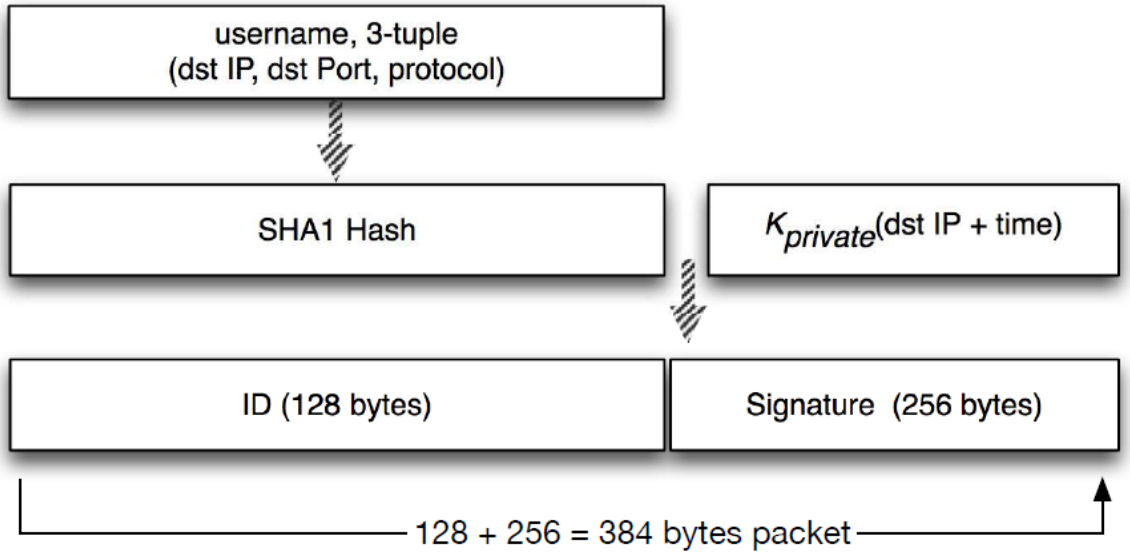


Figure 3.2: The NFC packet is composed by a connection identifier, ID, and the signature (destination IP/port and timestamp encrypted with the user's private key). ID is formed using the SHA1 key generated by hashing the subscriber's name concatenated with the 3-tuple information.

3.4 Privacy Preserving

NFCs preserve privacy or, more formally, a user who opts-in in NFCs does not receive degraded privacy compared to a non-NFC user. Since, NFCs are implemented at the network level, they do not interfere with technologies implemented at the application level. All anonymizing systems, such as Tor [18], can work in parallel with NFCs, and receive the exact guarantees they receive as of today. NFCs only verify to the last-mile ISP that the user is in fact connected to an anonymizing system, a piece of information that is already known to the last-mile ISP.

3.5 Traceability

NFCs do not express or force any policy for keeping logs. The only requirement is that the ISP keeps a signed record for any resource it serves. Depending on the esoteric policy used by the ISP, it may be possible that plain data is also kept, but hard to be taken advantage of, for any practical situation in the absence of the signed records. An ISP that wishes to protect users from a potential data leakage or from tracing attempts launched by third parties, can combine NFCs with a storage scheme that hardens revealing of the users' logs. The scheme is the following:

1. The ISP receives an incoming TCP-SYN packet and a signature for the new connection. The signature has been produced by encrypting the destination

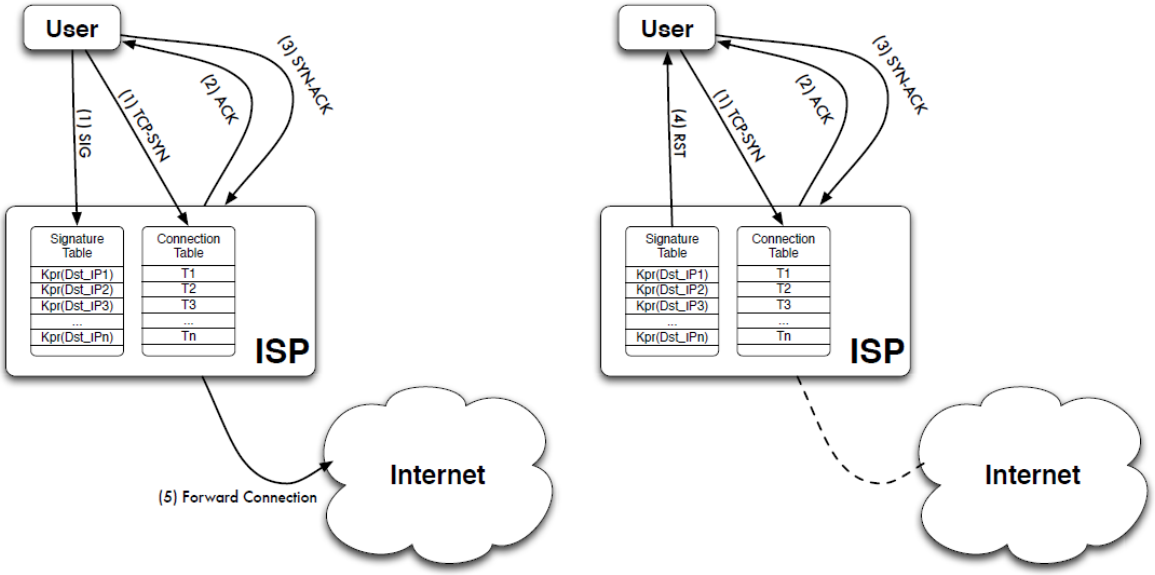


Figure 3.3: The NFC architecture. Every new flow is stored in the Connection Table and incoming signatures are stored in the Signature Table. If, after time T, no signature for a new flow is present in the Signature Table the new flow is terminated.

IP address of this new connection using the user's private key.

2. The ISP proceeds and decrypts the signature with the user's public key in order to verify it. If the result is indeed the destination IP address of this new connection the ISP keeps the signature. Otherwise it transmits a TCP-RST to the user's host, in order to terminate the connection.

3. Assuming that the signature is authentic, the ISP concatenates the current date and a random number, used as a salt and taken from a range of length R, to the signature. It finally, produces a cryptographic hash by hashing the result of the concatenation. The ISP deletes all information, including the salt, and keeps only the final cryptographic hash.

An attacker that has granted access to the ISP's logs and to the user's private key can only perform a brute force attack. If R is selected from a large space, then the brute force attack may be really inefficient. On the other hand, the signed records can be still of use. Assume that there is a case under investigation and there is a warrant issued, that allows a third party to query an ISP for the network profile of the suspect. Using the above scheme, the ISP by design is able to answer only if a particular subscriber has been connected with a particular host on a specific date. For example, the third party can form a query which is composed by the user, Alice, an IP address, Bob's machine, and a date D. This query is compiled

in: Did Alice connect to Bob's machine on D? In order for the ISP to answer this question the procedure below is followed:

1. Alice, the suspect, encrypts Bob's IP address with her private key and submits it to the third party.
2. The third party verifies that the produced signature is authentic by decrypting the result, which Alice submitted, with Alice's public key. If the product is indeed Bob's IP address, then the signature is considered authentic. If the signature is authentic, the third party submits the signature and the date D to the ISP for further processing.
3. The ISP takes the signature and concatenates the result with the date D. It then proceeds and generates the products of the concatenation of the final result with all possible numbers up to R
4. The ISP produces all cryptographic hashes by hashing all concatenations.
5. The ISP checks to see if any of the produced hashes exists in the storage, which contains all hashes associated with Alice's network activity and replies to third party positively or negatively according to the result.

The above procedure is required for answering one query. The time needed to complete the whole procedure depends on the chosen value of R and the algorithms used for the cryptographic hashing.

3.6 Scalability and Fault Tolerance

An ISP should take care that the NFC service is always in healthy condition. A failure will make all NFC-enabled subscribers incapable of accessing the Internet. However, NFCs require limited state and resources, namely keeping up to date the connection and the signature table, and verifying signatures. We can argue that both tables are not hard to maintain; the size of the tables are moderate. We collected some typical data from a large European ISP 2 . The ISP experiences tens of thousands of connections per second, while serving millions of mobile users. Assuming that a signature is 256 bytes and you need to store the signature for seconds (strictly you may store the signature for even less time of a second, as we discuss in Section 4, but keeping the signature in the table gives us a benefit from caching), then the storage required for the signature table is in the order of a few Gigabytes. Notice, that this storage is required for millions of subscribers accessing concurrently the ISP. Since, very limited storage is required, NFCs can easily scale. Additionally, the architecture is inherently parallelizable on a user basis. There can exist multiple independent NFC servers, working in parallel without any concern regarding integrity of the data. Finally, fault-tolerance can be easily achieved using standard techniques applied in databases systems and DHTs.

Chapter 4

Implementation

In order to implement the network flow contract in click modular router[47], we created new elements and modified preexisting elements to provide additional functionalities. Although click is enriched with a large number of elements, in the security sector there was no such a similar progress. Click modular router provides no elements for asymmetric cryptography. For the above reason we had to adapt the asymmetric cryptography algorithm from openssl to click modular router in kernel level. Another important area was the creation of kernel thread functions that are executed after a period T that is defined from the user. The third and last area was the creation of reset packets which are capable to terminate the flow connection.

4.1 RSA

In order to add the functionality of asymmetric cryptography in click modular router[47], we created a new element whose code runs on kernel mode. The cryptographic algorithm used is taken from openssl Eric Young version. The changes we have made concern the code itself, the algorithm remains the same. Indicative problems we encountered during adaptation were: access file to generate random numbers, the use of flow/double variable which is prohibited at click kernel level. The generation of random numbers was replaced with a secure random function that is provided from click. The implementation of the function that used flow variable was modified in a new function which maintains the same interface, precondition, post condition, but doesn't uses any flow variables. From the whole code of openssl, which numbered in tens of thousands lines, were filtered and maintained the functions that provide the below functionality:

- Creation of asymmetric RSA keys (public – private) based on input strings in format PEM (Privacy Enhanced Mail) which is a Base64 encoded.

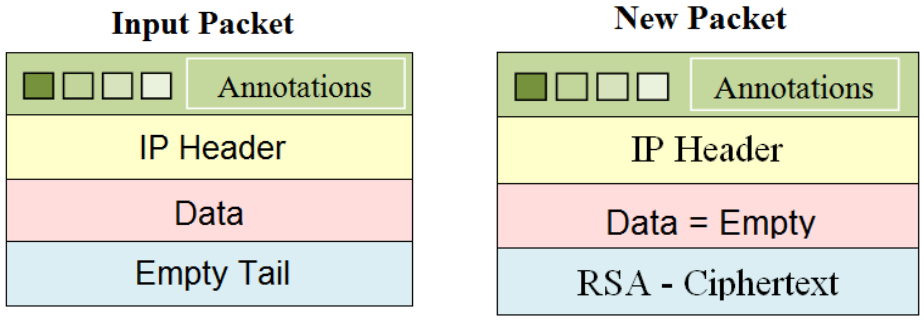
- Encrypt – decrypt functions that provide cryptography for the respectively

public, private keys.

The key used from the element is not given like an argument, but is indirectly received from the annotation region of the packet. Specifically, we have set as rule that the key that our element will use, will be determined from another element which makes the classification of the packets. According to the kind of packet, the corresponding key is determined. Our new element does not classify packets; its function is limited to encrypt/decrypt packets with keys that are defined in advance from another element.

The implementation of the element is stateless (i.e. no memory for packets it has seen before). In case of router fail, after router restart the element continues to work properly without any need to recover to the previous state. The stateless implementation is the basic principle for the normal operation of routers.

4.1.1 RSA Encrypt



(a) Input packet

(b) Output packet

Figure 4.1: Example of packet's encryption process from Rsa element

In client's router, the RSA element is configured to make encrypt. From each packet that is received from input zero, destination IP and destination port are extracted. This procedure is done with the function getSignData(Packet *p). The data of destinationIP| destination port | timestamp are encrypted with a client's private key, as shown in Figure 4.2 . The cipher text and the Packet p are given as parameters in the function sendSignature (const char * cipher,Packet *p). This function modifies the current TCPSYN packet to signed-packet. The new packet has also a unique ID that makes it different from the other TCP/UDP/ICMP etc. packets and identifies it as a signature packet. After that a tail Header is added where the cipherText-Signature is contained. After the completion of the above procedure and the set of checksum, the packet is forwarded to the next Element

as a normal packet and the function sendSignature() is returned. Figure 4.1 shows the packet's modification from RSA element during the procedure of encryption. Figure 4.1(a) shows the original TCP SYN incoming packet and Figure 4.1(b) shows the new packet as it outgoes from RSA Element. The outgoing packet is bigger because of the adding of cipher text to the packet's tail.

Figure 4.2: Data encryptions from Rsa element, with the two first being extracted from the packet and the third one from the system.

4.1.2 RSA Decrypt

n ISP router, the RSA element is configured to make decrypt. In case that the element RSA makes decryption, it extracts from the packet's annotation area the RSA key that it will use. Then it extracts from packet the cipherText located in the packet's tail. The cipher text is decrypted with a public key and the resulting plainText is overwritten to the packet's tail and the new modified packet is forwarded to the next element. In Figure 4.3 we present the push function of Rsa element that modifies the input packet.

4.2 SPI_ESPEncap

The element was created to add to the packet an extra header that includes within the spi number, which corresponds to the key that the packet was signed. Specifically, this element takes from the packet's annotation place the spi and thereafter adds to the packet's head a new extra header that includes within the spi number. This header will be used from the next router so that it can find the key with which the packet will be decrypted. The new packet is forwarded to the output port zero. This element has no arguments.

4.3 SPI_ESPUnencap

The element was created for the reverse process of SPI_ESPEncap. Specifically, it removes the additional header from the packet's head and the modified packet is forwarded to the output port zero. The field SPI, that is contained in this header has been used from element RadixIPrsaLookup, which extracts the SPI number and sets it in a specific area of the packet's annotation that click modular router provides.

```
void rsa ::push(Packet *p){

   rsa_key = RSA_SA_REFERENCE_ANNO(p)
   clen = RSA_size(rsa_key);
   switch(_op){
     case ENCRYPT :
       ptext = getSignData(p);
       plen  = strlen(ptext);
       if(_opPub_Priv == PRIVATE)
         RSA_private_encrypt(plen,ptext,ctext,rsa_key,RSA_PKCS1_PADDING);
       else if(_opPub_Priv == PUBLIC)
         RSA_public_encrypt(plen,ptext,ctext,rsa_key,RSA_PKCS1_PADDING);
       sendSignature(ctext,p);
       output(1).push(p);
       return;

     case DECRYPT :
       const u_char *ctext = p->data()+p->length()-clen;
       if(_opPub_Priv == PRIVATE)
         RSA_private_decrypt(clen,ctext,ptext,rsa_key,RSA_PKCS1_PADDING);
       else if(_opPub_Priv == PUBLIC)
         RSA_public_decrypt(clen,ctext,ptext,rsa_key,RSA_PKCS1_PADDING);
       checkSignatureList(ptext);
       output(1).push(p);
       return;
  }
```

Figure 4.3: Implementation of Rsa push function

4.4 RSA Client Manager

The RSA Client Manager is the element that manages the RSA Key. The element has two arguments:

- string RSA key, in format PEM (Privacy Enhanced Mail) which is a Base64 encoded.
- integer SPI.

SPI is a unique number that identifies the RSA pair key that the user uses. Specifically, when a packet contains a signature and is forwarded to the other side, then for the server the following issue arises: which key have to be used for the packet decryption? In order to solve this problem we have defined that the client and the server have agreed that each RSA Key corresponds to a unique SPI number. This number is forwarded with the signature as an additional Packet's Header.The element RSA Client Manager adds to the specific annotation area of the packet the RSA key and the SPI in order to make them available to other elements in the

```
clMan::rsaClientManager(
»       »       »       "-----BEGIN RSA PRIVATE KEY-----\n\
»       »       »       MIIBOwIBAAJBAK63mpiqFFEO0Bb1PIndu19i/WQ8B6+kXbMz8xTYFxS+/V8KsHUP\n\
»       »       »       1Rr9FS+SuJpjcIGD1anbfKmzZ/W/vLoXRNcCAwEAAQJAeSHIaEG61jv7337lGvyE\n\
»       »       »       1Owvm9fWyFqfsAe7Z18uGW3vjQUFXeAe4jpHY2lKh5bXYwFyPvpq9b8FJlE2PzUz\n\
»       »       »       oQIhANt+uowUUYuuY5OksIdub/ks+92ExWVnc2m/RwAZZ3ixAiEAy8ZnZdYZtVKE\n\
»       »       »       o6L7lHr2bgpRl6yULautQGPQPO6weAcCIQCerxwGsZnSthzfXK0S/FCPgGoGWDqt\n\
»       »       »       eQEDbnnSJIudwQIhAILVakHKS0tOJfdih6D3qg1Q9r4pgiIond4/t83Heuo1AiBc\n\
»       »       »       w7dCpyPe0V8E++ksQMdzdA6XQZA4qAD/iJN6UR7CSw==\n\
»       »       »       -----END RSA PRIVATE KEY-----\n", 123);
```

Figure 4.4: Rsa Client Manager configuration.

current router. After that, the packet is forwarded to the next element.An example of rsaClientManager element is shown in Figure 4.4

4.5 RSA Client Lookup

RSA Client Lookup is the element that creates a copy of each TCP SYN packet and forwards it to output port one. These packets are intended to be signed. The rest packets are forwarded normally to outputs which correspond to packet's destination IP. This element constitutes an extension of RadixIPLookup which is described below.

4.5.1 RadixIPLookup

Performs IP lookup using a radix trie. The first level of the trie has 256 buckets; each succeeding level has 16. The maximum number of levels that will be traversed is thus 7. Expects a destination IP address annotation with each packet. Looks up that address in its routing table, using longest-prefix-match, sets the destination annotation to the corresponding GW (if specified), and emits the packet on the indicated output port. Each argument is a route, specifying a destination and mask, an optional gateway IP address, and an output port. The interface that is provided to the user is represented below.

- RadixIPLookup(ADDR1/MASK1 [GW1] OUT1, ADDR2/MASK2 [GW2] OUT2, ...)

4.5.1.1 IPRouteTable

IPRouteTable defines an interface useful for implementing IPv4 route lookup elements. It parses configuration strings and calls virtual functions to add the resulting routes. A default push function uses those virtual functions to look up routes and output packets accordingly.

4.6 RSA Server Lookup

The packets are forwarded normally to outputs which correspond to packet's destination IP. If a packet has a TCPSYN flag, than a copy of this packet is forwarded to the output port zero, which is connected with element sign-Manager which is described below. If the packet's id belongs to the signature packets, then the packet is forwarded to the output port one.

4.7 RSA Server Manager

The RSA Server Manager is the element that manages the RSA Key. The element has a list of the below arguments:

- string RSA key, in format PEM (Privacy Enhanced Mail) which is a Base64 encoded
- integer SPI

The element stores the above list of arguments in a map data structure, where the index is the SPI. The element checks for each packet the specific header that contains the SPI. Based on this number searches on its map data structure which RSA key corresponds. Afterward stores to the specific packet's annotation area the corresponding RSA key in order to make it available to other elements in the current router. After that, the packet is forwarded to the next element. An example of rsaServerManager element is shown in Figure 4.5

```
servMan::rsaServerManager(
»       »       »       "-----BEGIN PUBLIC KEY-----\n\
»       »       »       MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAK63mpiqFFEO0Bb1PIndu19i/WQ8B6+k\n\
»       »       »       XbMz8xTYFxS+/V8KsHUP1Rr9FS+SuJpjcIGD1anbfKmzZ/W/vLoXRNcCAwEAAQ==\n\
»       »       »       -----END PUBLIC KEY-----\n", 123);
```

Figure 4.5: Rsa Server Manager configuration.

4.8 Sign Manager

The Sign Manager Element receives as input only two types of packets:

- TCPSYN packet
- Signature packet

The output packets are TCPRST packets. Specifically, as soon as the element receives a TCPSYN packet extracts from it the destination IP, destination Port

and then takes a current timestamp. Based on these creates a kernel thread Timer that executes its reset function after a period T. If the Timer is executed, then a reset function creates TCPRST packets to terminate the flow connection.

The element receives signature packets, whose signature is already decrypted and the plaintext is stored in the packet's tail. The element checks the signature id with Timer's id. If there is a match, then the corresponding thread is terminated.

Chapter 5

Evaluation

In this section we will describe an experiment where a client router and an ISP router are simulated. In this connection we offer the client the possibility to identify the flows he uses. Specifically, in this new connection that we create all the TCP SYN packets are signed with the user's private key. When the ISP router receives TCPSYN packets, then it waits for a period of time T to receive the signature for the corresponding flow. If the time expires, then the ISP router creates a burst of reset packets to terminate the flow.

A summary of the packet processing steps at the client (Left) and ISP router (Right) is shown in Figure 5.1. We see that directed graph of processing elements and their interconnection according to how packets flow between elements.

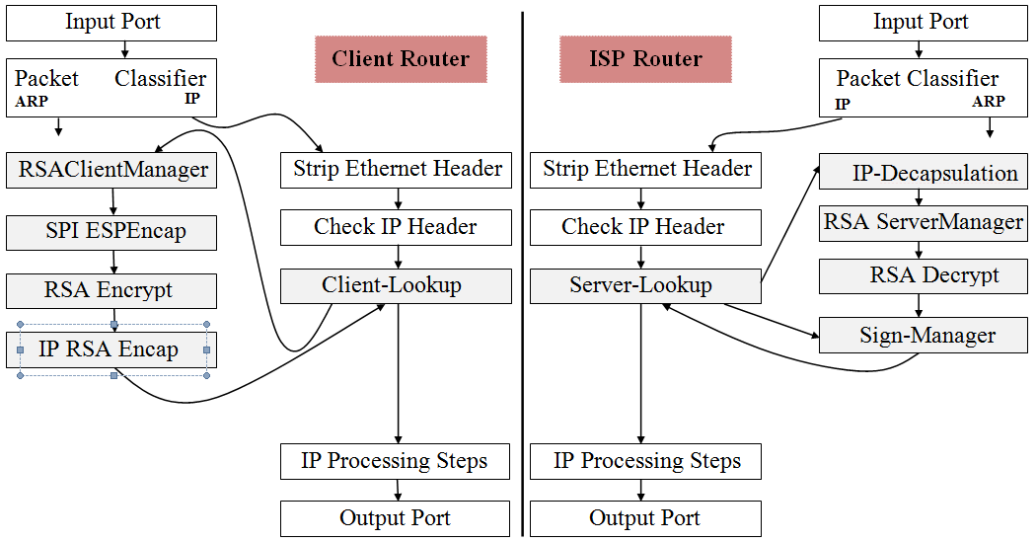


Figure 5.1: processing path at Client and ISP router.

5.1 Client router

Initially, the incoming packet contains an Ethernet header which is removed from element Strip. Then with the use of CheckIPHeader, the packet's Checksum is controlled and repaired. This process is showed in Figure 5.2. The packet is forwarded toward ClientLookup, which will make the classification based on packet's type and its destination IP. According to these parameters ClientLookup will forward the packet on the corresponding process path. In case that an incoming packet is a TCPSYN packet, then ClientLookup creates a copy of this packet and forwards it to the output port zero. At this point begins the process where the packet is processed in order to be signed. The element RSA Client Manager adds to the specific annotation area of the packet the RSA key and the SPI in order to make them available to other elements in the current router. An example of RSAClientManager is showed in Fig. 4.4. After that, the packet is forwarded to the next element. The element SPI_ESPEncap takes the SPI number from a specific area of the packet's annotation place. The SPI number is added as a header on the packet's head. This header will be used from the ISP router to find the RSA key useful for the packet's decryption. The new modified packet is forwarded to output port zero. The next element used in this process path is RSA Encrypt. The arguments defined for this process are encryption with a private key. RSA Encrypt element takes the private key from the packet's annotation place. The RSA key as we have already mentioned above is placed from RSAClientManager to the packet's annotation place. The RSA Encrypt element extracts from the current packet the destination IP and the destination port. Both these data are concatenated with a timestamp and the result is encrypted with a private key. The cipher text that results is added as an additional header to the packet's tail. The modified new packet is forwarded to the output port zero. Finally, the last element that modifies the packet at network level is IPsecencap. The element adds to the packet's head an IPheader which corresponds to ISP router IP. This header, except the basic information that an IPheader should have e.g. destination IP, destination port, source IP, source port, time to live (TTL) etc., defines an additional protocol number that corresponds to our defined protocol for asymmetric identification. Once the successful addition of this header is completed, the packet is forwarded to output port zero. In Figure 5.1 are shown all the elements that take place in the above procedure.

```
// Hand incoming IP packets to the routing table.
// CheckIPHeader checks all the lengths and length fields
// for sanity.
ip :: Strip(14)
    -> CheckIPHeader(INTERFACES 192.168.177.24/24 192.168.178.11/24)
    -> [0]rt;
```

Figure 5.2: The removing of ethernet header and the repairing of packet's checksum.

5.2 ISP router

The received packet is already encapsulated with an Ethernet header which is also removed from element Strip. After that, CheckIPHeader calculates the packet's Checksum and sets it to the packet's Checksum field. The packet is forwarded toward ServerLookup. Before forwarding the packet on the corresponding process path, this element makes the classification of the packet based on its type and its destination IP. In case that the received packet's IPheader PROTO field has the value 50, then we recognize that the packet has been signed and it is forwarded to output port zero. Output port zero is connected with the element IP-Decapsulation. In case that the received packet is a TCPSYN packet, then a copy of the packet is created, which is forwarded to output port one. Output port one is connected with the element Sign-Manager. The rest of the packets are forwarded to the IP Processing Steps.

The next element that manages the packet, after ServerLookup is IP-Decapsulation. This element removes the IP header that we had manually added at the client router. The element RSAServerManager extracts the SPI number from the packet's head. Based on this number it searches on its RSA_sa_Table structure in order to find the key that is responsible to verify the packet's signature. This RSA key is placed from RSAServerManager to the packet's annotation place. This procedure is done so that the other elements involved in this process can have access too. Afterwards the packet is forwarded to output port zero.

Then RSADecrypt element assumes the processing of the packet. The arguments of RSA element are defined to make decrypt with the public key. The element takes the public key from the packet's annotation place and the cipher text from the packet's tail. Based on the key and the cipher takes calls the decrypt function. The plain text that results from decryption has the below format:

- destIP | destPort | timestamp

The resulting plainText is overwritten to the packet's tail and the new modified packet is forwarded to the next element. The Sign-Manager element as described above holds a kernel thread Timer for each TCPSYN packet. When it receives a signature packet, whose signature is already decrypted and the plaintext is stored in the packet's tail, then the element checks the signature id with Timer's id. If there is a match, then the corresponding thread is terminated. The Sign-Manager element receives TCPSYN copy packets from the ServerLookup and based on that creates a Timer kernel thread for each packet. The above process is shown in Fig. 5.1

5.3 T - threshold Estimation

For each new network flow introduced by the subscriber (client), the ISP expects to receive a valid signature that verifies that the user agrees by a digital contract

to reach the particular destination. The ISP waits for the signature up to a time threshold, denoted as T through the thesis. Notice, that this is not an overhead. The TCP handshake proceeds as normal, but, in case the signature has not been received after time T, the server terminates explicitly the connection by sending a TCP-RST.

In order to estimate typical values of T we proceed and carry out the following experiment. The client generates 100,000 TCP-SYN requests towards the server, using the hping3 tool in fast mode (10 requests per second). We record the followings:

- The time needed for the client to create a signature (this is the Crypto part).
- The time elapsed from the arrival of the TCP-SYN packet to the arrival of the signature, as recorded by the server (this is the Overall part).
- The RTT as reported by hping3.

We conduct the experiment for two configurations. The first configuration is composed by two Linux desktops connected through a Fast Ethernet (FE) network with 2048 RSA key size. The second configuration is composed by the same above configuration with 1024 RSA Key size. We choose these configurations for the following reasons. As far as the FE experiment is concerned, where network overheads are negligible, we select the client and the server to be an normal PC (The client and Server machine is a 4 years old Linux box)with moderate computational capabilities, since we focus on the overhead imposed by the cryptographic operations. We depict the results for the FE with 2048 RSA keysize case in Figure 5.3 and in the Figure 5.4 the results for the FE with 1024 RSA key size. We plot a histogram for the average inter-arrival time of the TCP-SYN and the signature, and the average time spent in cryptographic operations. We also plot the CDF of RTTs; the distance between the beginning of the positive part of x- axis and the Crypto bar is 1. A first look suggests that the signature arrival time is greater than a typical RTT value. This is notably because of the computational part associated with cryptographic operations. To support this claim, in Figure 5.3 we also plot a second graph zooming in the top of the histogram. Notice, that the majority of the time is spent on cryptographic operations and the network lag, which involves the inter-arrival of the TCP-SYN and the signature packet at the server, is in the scale of the RTT as reported by hping3. Notice, also, that if the standard deviation (depicted in the plot with the vertical error bar) associated with the crypto computation is accounted, the crypto part can exceed the overall time required for the two network packets, namely the TCP-SYN and the signature, to arrive at the server. This is because of buffering in the network driver of the client causing the TCP-SYN packet to leave the host after the cryptographic computation has started.

In Figure 5.5(a) is shown the time that the client takes to sign a packet with 1024 bits RSA key and in Figure 5.5(b) present the times that ISP server takes to decrypt the signature . Also in Figure 5.6(a) is shown the time that client takes to sign a packet with 2048 bits RSA key and in Figure 5.6(b) present the time that ISP server takes to decrypt the 2048 bits signature.

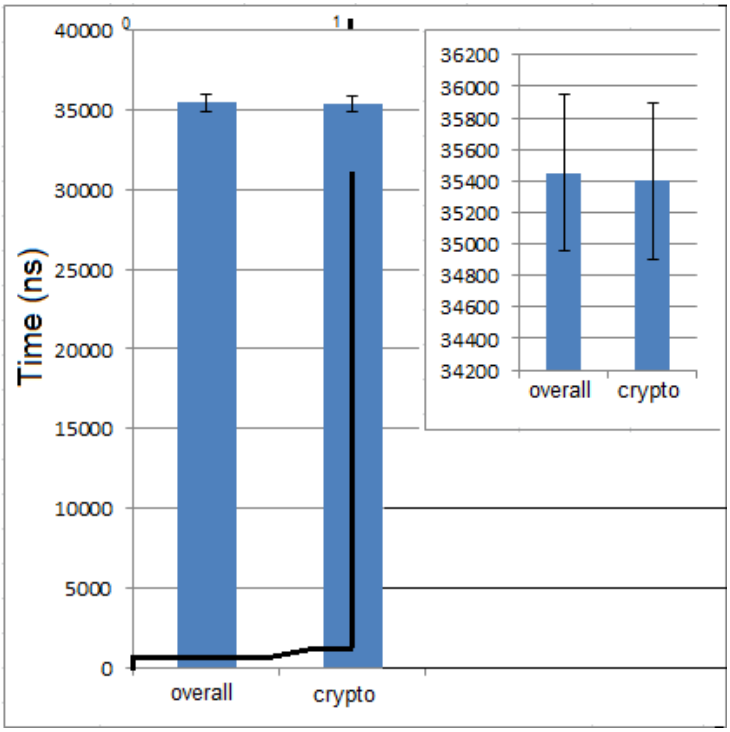


Figure 5.3: Evaluation for an NFC client 2048 RSA Key connected with an NFC server through a Fast Ethernet network.

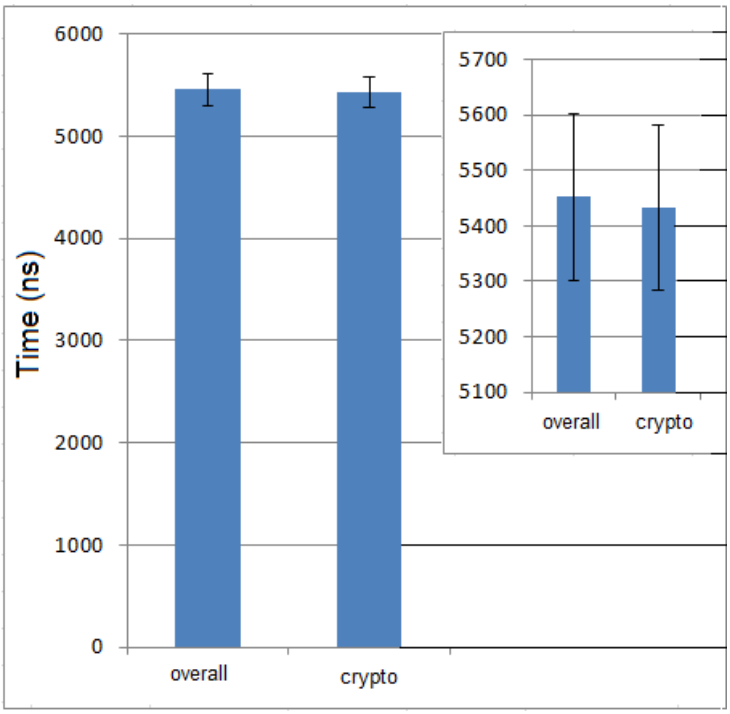
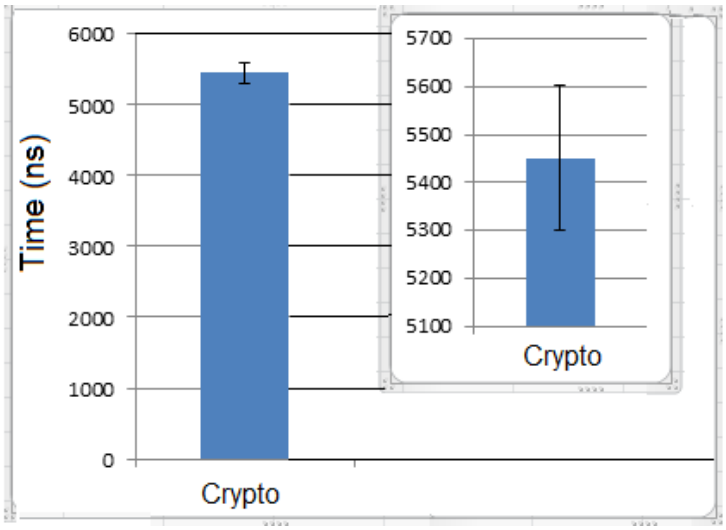
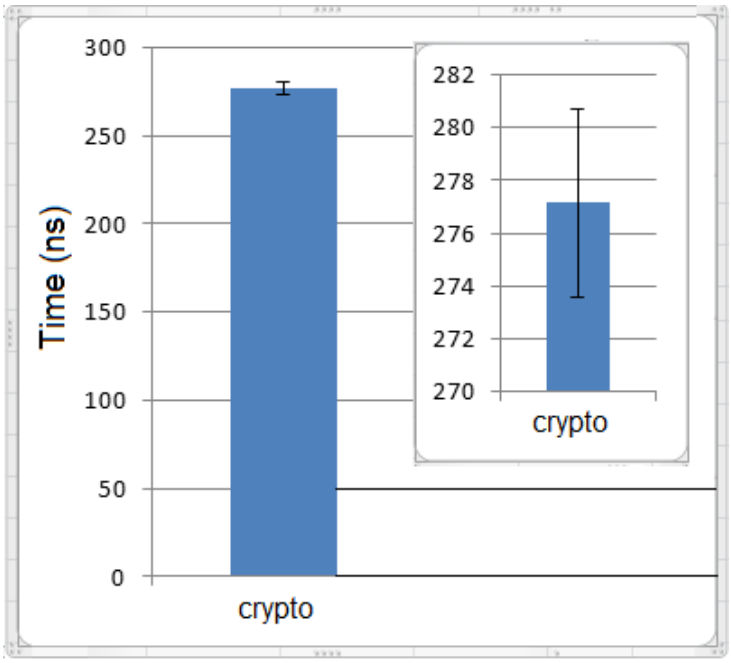


Figure 5.4: Evaluation for an NFC client 1024 RSA Key connected with an NFC server through a Fast Ethernet network.

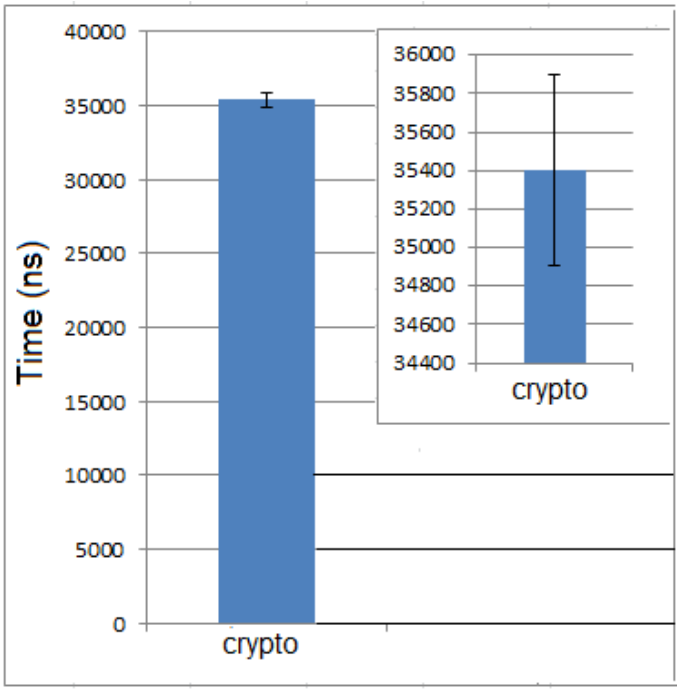


(a) Encrypt with private 1024 bits RSA key

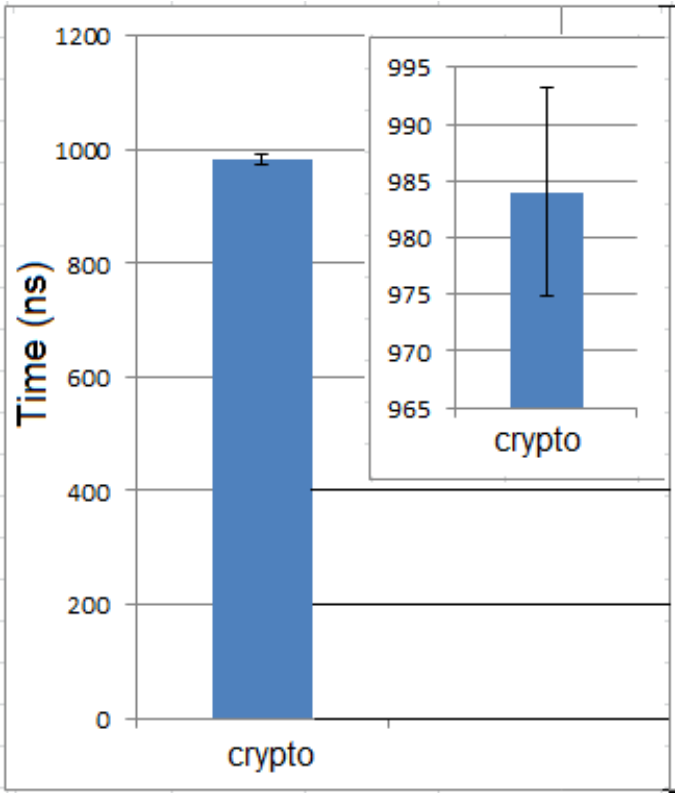


(b) Decrypt with public 1024 bits RSA key

Figure 5.5: Encryption - Decryption process with 1024 bits RSA Key



(a) Encrypt with private 2048 bits RSA key



(b) Decrypt with public 1024 bits RSA key

Figure 5.6: Encryption - Decryption process with 2048 bits RSA Key

Chapter 6

Related Work

Although this field of research is not new, there are a few recently published research efforts, which share many common properties with NFCs. We now discuss the most relevant to this Thesis works and highlight all factors that differentiate NFCs from similar proposals.

6.1 Pretty Good Packet Authentication

Pretty Good Packet Authentication (PGPA)[24] is a proposal that shares similar goals with this thesis. In PGPA, the authors propose a system that hashes all user's outgoing traffic and stores it to a device. The device must be placed over the access link used by the user to connect to the ISP. It can be placed towards the user's side, i.e. in the user's household, or it can be hosted by the ISP. Both choices share advantages and disadvantages. NFCs differentiate from PGPA in the following way. First, PGPA is proposed as an all-in solution for both chasing attackers that produce offensive traffic, as well as protecting victims that are wrongly accused for having offensive network activity. On the other hand, NFCs is an opt-in approach and focus only on protecting innocent users for incrimination stemming from their network activity. The difference between all-in and opt-in is significant. For example, PGPA fails to pinpoint potential attackers if the device is hosted in the user's premises, since a user may simply turn it off. The authors claim that turning off such a device may be evidence for criminal activity. We believe that this is questionable. The device can simply stop working due to a failure. How can someone judge if the device's failure has been caused by a user's explicit action or not? Second, PGPA stores all hashes for all packet's payloads. We agree that commodity storage is cheap nowadays, but, still, consider the case where a user is mobile. Today, there are more than one billion smartphone devices. Many of them are used for accessing the Internet using a data plan offered by an ISP. In that particular case, NFCs can protect the mobile user. On the other hand, PGPA requires a dedicated device in between the user's access link and the ISP. Since, a mobile user switches access links, then placing such a device is hard. A possible

solution, would be to implement the device on the smartphone, but, unfortunately, smarthphones have still very limited storage capabilities. Another possible solution is to place the device at the ISP's premises. In our threat model the ISP is considered trusted but not totally secure, thus the device, if stored at the ISP's level, can be modified by an inside or external attacker. Also, as the PGPA authors suggest, placing the device at the ISP side is not considered scalable.

6.2 Packet Attestation

Packet Attestation (PA)[9] suggests installing special mon- itors, independent of the current Internet's routing infrastructure, in every Autonomous System (AS). These monitors hash and collect all user traffic, and they can attest if the traffic is authentic under a particular case, where a user is accused for receiving offensive traffic. PA monitors, in the digital world, serve as an analog to eyewitnesses of the real world. PA needs significant deployment effort for installing all monitors and enough storage for hosting all traffic. It is true, as the authors point out, that the deployment can be done incrementally and storage needs are not unrealistic if the traffic is stored for a short period of time, such as a month. NFCs can also be deployed incrementally, and, also, a user needs to find just one NFC-enabled ISP to receive protection. The major difference between NFCs and PA, is that PA requires that the ISP does not collude with an attacker. NFCs are also limited against a threat model, where an ISP is completely rogue, but they can handle common cases where a legitimate ISP suffers from an insider threat or can be partially compromised. In that case, an attacker can plant fake information in the ISP logs and trap an innocent user.

In summary, although we believe that, both PGPA and PA, are designed towards the right direction, NFCs attempt to solve a more limited and narrow threat-model, which, nevertheless, can lead to severe problems, and, thus they are more practical and easier to deploy in the current Internet.

6.3 Privacy Preserving Network Forensics

Clue [7] attempts to bring the notion of physical evidence, such as DNA, in the digital world. More precisely, Clue uses Group Signatures[13] for delivering a scheme, where each Internet packet can be attributed to its original source. This scheme can assist in network forensics, without sacrificing much of the user's privacy, as it is experienced with current technologies. Using Clue attackers can be traced even years after their committed crimes. Clue is targeting solely the investigation of actual crimes and how to chase attackers to their source. On the other hand, NFCs do not aim at chasing cyber-criminals but at protecting innocent users that are wrongly accused. In a world, where Clue has been deployed and there is perfect packet attribution, our threat model does not exist. However, we believe that we are far from actually deploying a system such as Clue. Currently, there is still no

practical framework for protecting innocent users from incrimination through data fabrication.

6.4 Internet-wide Systems

Accountability in the Internet has been the objective of an ongoing research. Scientists have described complete systems that, if applied, can make the Internet accountable. This will further assist in building easily technologies for addressing a large number of problems, such as source spoofing, route hijacking and forgery, or Denial of Service (DoS). PI[45] , AIP[8] and Passport[33] are three systems towards such a direction. In a similar fashion, researchers have pro- posed a Framework for Internet Innovation (FII)[30] , which will allow and promote innovation of the current Internet. This innovation will be able to eliminate many deficiencies that current Internet experiences. Among many things, the authors of the FII propose accountability features.

All these systems propose major reconstruction of fun- damental parts of today's Internet. We agree with a large fraction of these proposals and we believe that a clean-slate design will terminate a rich collection of problems that we exhibit today. However, deployment is not always trivial. Changing fundamental Internet-core concepts, such as, for example, routing or addressing, is really hard. On the other hand, Internet, as it is currently designed, experiences prob- lems, which can have serious consequences. One such a problem is the one we are dealing with in this problem. Until these clean-slate approaches reach the level of maturity and start getting deployed in the wild, solutions such as NFCs, which can be deployed immediately, can be of use.

6.5 Scalability and Fault Tolerance

Plenty of research for privacy has been associated with network activities performed by end users. There is a huge effort for designing and implementing systems for providing anonymous communications and some of them have lead to concrete systems that exist in the real world. In this thesis, we review privacy from a different perspective. Instead of providing the design of a system that hides the network traces of a user, we propose a technology that tracks and cryptographically proves all network activity between the user and the last-mile ISP. However, NFCs are implemented at the network level. All currently employed anonymizing systems are developed at the application level and, thus, there is no interference between NFCs and anonymizing systems. A user, who is utilizing concurrently NFCs and an anonymizing system proves to her lastmile ISP, that, in fact, she is using an anonymizing system; information who is already known by the user's ISP.

The idea of providing authenticity and confidentiality of network traffic is not new. There are concrete efforts for standards towards this direction, such as IPSec . Although the computational overhead of encrypting all traffic is currently re-

strictive for a massive deployment, researchers have produced fast and practical implementations for encrypting TCP traffic . In this thesis we seek a practical and efficient solution for a very precise problem. Instead of producing a very elaborate cryptographic system, we provide a fast implementation that performs cryptographic signing of every TCP-SYN packet, in order to allow users accounting for their network activity, by verifying all traces they leave in the last-mile ISP.

Chapter 7

DISCUSSION AND FUTURE WORK

In this section we discuss in details various points that can clarify the behavior of NFCs under potential misusing and attacks, as well as various issues that need a more in depth analysis. We also present our plans for future work.

7.1 Replay attacks

As we discussed in Section 3 an NFC signature is constructed by hashing the destination IP/port, the user is willing to access, concatenated with a timestamp. This is essentially for protecting the user against replay attacks [42]. More precisely, it is true that IP addresses can change in numerous ways. First, an IP address may change owner and, thus, the content associated with this particular address may become substantially different from the one existing at the time the user accessed the resource. Second, the machine associated with a particular IP address can be compromised and start serving offensive content. Omitting the timestamp from the signature allows an attacker to replay old signatures, which describe hosts that may have changed at a later time than the one were originally accessed. Timestamping the signatures can protect the users against such replay attacks.

7.2 IP Transitivity

One can argue that signing IP addresses raises concerns. It is common for many host providers to use virtual hosting or Cloud providers to group many resources in a single IP. Thus, an IP address and a port number cannot describe accurately what the user has actually accessed. However, implementing NFCs at the network-level allows us to build a generic solution for practical accountability without incorporating in the framework any semantic information from the application level, which sometimes is hard to compile. Using NFCs we can record user accesses towards any service and not just web surfing.

We speculate that the IP address will be a more accurate descriptor of a host in the near future. This is mainly driven by the fact that IPv6 [16] has started to be deployed by many Internet providers and all modern OSes are now IPv6 enabled. Eventually, we believe that IPv4 will be replaced by IPv6 and, then, IP transitivity will be greatly reduced.

Moreover, even in current Internet, there are protocol proposals, which can greatly assist in a more accurate operation of NFCs. TCP Fast Open (TFO) [35] is an attempt to speed up content delivery in clients that communicate over TCP by reducing extra RTTs. TFO suggests that the first TCP packet of the flow can be encapsulated in the TCP_SYN packet. Notice, that attaching data in a TCP_SYN packet is perfectly legitimate according to RFC 793 [27]. Moreover, attaching the first packet to the TCP_SYN packet greatly speeds up application-level protocols such as HTTP, which usually are based on short lived sessions involving just one GET/POST request. With TFO in place, NFCs can be very easily extended to sign more accurate descriptors than the IP/port pair. Instead of using the IP/port information of the destination host, signer calculates the digest of the TCP_SYN packet, creates the signature and sends it to the ISP. Notice, that now the TCP_SYN packet includes the first TCP packet which has application-level info - for example, in the HTTP case, the packet includes the URL the user is trying to access. On the other hand, the ISP will check if the digest is valid and forward the packet. Notice, that a fast symmetric algorithm can be used for calculating digests. Incorporating TFO with NFCs is something we plan to investigate in our future work.

7.3 ISP Incentives

It is questionable why an ISP should want to design and implement an NFC service. We believe that the rising number of security incidents and host compromising can lead ISPs to promote security packages, which include many security services combined, such as malware detection, SPAM filtering, Botnet detection, etc. Many of these services have implicit benefit for the ISP. First, users that enjoy such services form a less hostile group of hosts that must be orchestrated by the ISP. Compromised hosts can infect and perform damage to hosts participating in the same network. Second, by offering security services the ISP is more appeal- ing to their subscribers. Offering NFCs as a service can make them more attractive and more competent in the ISP market. And third, as NFCs directly are concerned, they can offer, implicitly, added security benefits to all subscribers. Consider that an NFC user must encrypt each TCP_SYN with her private key for connecting to the Internet. Third parties that can penetrate the wireless router of the subscriber, either because there is a bug in the router, the security is weak or the set-up was purely configured, cannot steal her Internet, since they have no access to the key.

7.4 UDP Traffic

NFCs are applied to TCP traffic. It is believed that the vast majority of Internet applications use TCP [11]. However, UDP traffic is increasing, since popular applications for VoIP or BitTorrent are heavily based on UDP. In contrast with TCP, UDP is unreliable, meaning that there is no state at the network level. Consequently, there is no handshake at the network level and, thus, the NFC client cannot easily capture the starting of the flow. However, protocols that emulate state purely in UDP do exist. NFCs can be applied also for UDP traffic. The signer software can be extended to passively monitor UDP traffic and capture every outgoing UDP packet, which is followed immediately by a UDP response, indicating the existence of a possible handshake in UDP.

Another problem with UDP, is that you cannot explicitly terminate the exchange of UDP packets by sending a specific packet (i.e., as is TCP_SYN for TCP). You need to know the application logic for figuring out how is the UDP conversation terminating. Nevertheless, the ISP could simply drop all UDP packets exchanged if there is no valid signature received. The technique for handling UDP, in principle, is very similar to TCP. We leave handling of UDP traffic for future work.

7.5 RSA Cost

Traditionally, public-key cryptography is computationally expensive. Throughout this thesis, we have presented various measurements regarding the computational overhead imposed on signing and verifying a signature. As far as the client is concerned, we argue that RSA encryption does not impose any dramatic overhead that can alter the user's perception. As we showed in Section 4, even a low-power netbook needs nearly 50 ms to compute a signature with a 2048-bit key. RSA encryption can be rather expensive for mobile devices, such as smart-phones. We leave this for future work. As far as the server is concerned, hard- ware acceleration can be of great assistance. Consider that a Sun's UltraSPARC T1, equipped with a Modular Arithmetic Unit for RSA, could perform 20,425 signature verifications per second using a 2048-bit key utilizing all 32 cores, 7 years ago [6]. Decryption can be further speed up by using GPUs [29] or modern x86 CPUs, which have implemented operations needed by cryptographic algorithms directly in hardware and can accommodate encryption at line speeds [31]. Encryption and decryption at line speeds is no more an issue has been also argued by other researchers [11].

7.6 URL Inline Linking

Inline linking, or hot-linking, is when a web site directly links objects, like images or frames, hosted at third parties. A malicious web site can trap a user, by inline linking offensive resources in hidden frames. Since, all signatures are generated

automatically, accesses for the offensive resources will be also signed. Although inline linking is a fundamental property of the web platform, it has been shown that can be greatly abused in a variety of ways [32]. Attacks of this type are out the scope of our threat model. However, there are concrete efforts towards technologies, which prohibit the in- line inclusion of resources hosted at third parties. Mozilla is leading this effort, which is considered as the state-of-the-art in browser security [40].

Furthermore, we strongly believe that for a user to be concretely accused for accessing offensive resources, a rich profile must be synthesized. Even if a user is lured to visit a particular web site, which redirects all visitors to offensive resources, there is still a way for the victim to prove her innocence. First, the user can prove that she first visited the malicious web site and then the offensive resources. Second, the user can claim that she wasn't ever associated with offensive resources in the past. In order to accuse a user for accessing offensive resources, someone needs to create a behavior experiencing a particular pattern, associated with visiting offensive resources, regularly. This profile can be easily created through data fabrication if an attacker has access to plain ISP logs, but computationally hard if NFCs are in place.

7.7 Rogue ISPs

In the case the ISP is malicious, then it can explicitly inject offensive traffic towards the subscriber and force her to implicitly sign it. For example, the ISP could inject in web pages JavaScript, which generates HTTP requests to- wards offensive resources. We consider that in the general case an ISP will not be malicious, since offending their sub- scribers is not compatible with the ISP's business model. In the past we experienced the existence of rogue micro-ISPs, but still their operation was short. They were immediately shut down, by the time their malicious activities were dis- covered [17]. The threat model we define in this thesis does not address the case of a malicious ISP. However, we can address the case of a legitimate ISP with a malicious in- sider [41], which we consider far more realistic.

Chapter 8

CONCLUSION

In this thesis we designed and evaluated Network Flow Contracts (NFCs), a practical and lightweight accountability framework for authenticating all network accesses at the ISP level. An ISP providing NFCs can guarantee that the network activity of each subscriber has been logged with her consent. In case an NFC-compliant ISP is compromised, data fabrication is computationally hard, since all network accesses are signed with the user's private key. More precisely, each initiating network flow is explicitly signed by the subscriber. The ISP terminates all connections for which no valid signature is received after a time threshold. This does not introduce any extra lag to the client's connection, since the TCP handshake proceeds normally and, in case there is no valid signature, it is terminated asynchronously. NFCs do not degrade the user's current privacy. To the contrary, NFCs are implemented at the network layer and, thus, they do not interfere with anonymizing systems, such as Tor [18], which are implemented at the application layer.

Finally, in this thesis we proposed a novel storage scheme for keeping all network accesses securely at the ISP. In case the ISP is compromised, the attacker cannot reveal the network history of any subscriber of the ISP. In case an attacker compromises the ISP and the user's private key, it is still hard to reveal all network history of the particular user. Nevertheless, the encrypted logs can be still of use under a crime investigation, if the suspect agrees to provide specific resources encrypted with her private key.

Bibliography

[1] *Anonymous free speech is protected by the First Amendment. https://www.eff.org/issues/anonymity.*

[2] *Directive 2006/24/EC on Data Retention. Official Journal L105, 13/04/2006 P.0054-0063. http://eur- lex.europa.eu/LexUriServ/LexUriServ.do?uri= CELEX:32006L0024:EN:HTML. January 2012.*

[3] *H.R.1076: Internet Stopping Adults Facilitating the Exploitation of Today's Youth (SAFETY) Act of 2009. http://www.govtrack.us/congress/bill.xpd?bill= h111-1076.*

[4] *IRCache. http://www.ircache.net/.*

[5] *Opinion of the European Data Protection Supervisor. http://www.edps.europa.eu/EDPSWEB/webdav/site/mySite/shared/Documents/Consultation/C 05-30-Evaluation-Report-DRD-EN.pdf.*

[6] *RSA Performance of Sun Fire T2000. http://blogs.sun.com/chichang1/entry/rsa performance of sun fire. January 2012.*

[7] *M. Afanasyev, T. Kohno, J. Ma, N. Murphy, S. Savage, A. C. Snoeren, and G. M. Voelker. Privacy-preserving network forensics. Communications of the ACM, 54:78–87, May 2011.*

[8] *D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). In Proc. ACM SIGCOMM, Seattle, WA, Aug. 2008.*

[9] *Andreas Haeberlen, Pedro Fonseca, Rodrigo Rodrigues, and Peter Druschel. Fighting Cybercrime with Packet Attestation.*

[10] *BBC. Email and web use 'to be monitored' under newlaws. http://www.bbc.co.uk/news/uk-politics-17576745. April 2012.*

[11] *A. Bittau, M. Hamburg, M. Handley, D. Mazieres, and D. Boneh. The case for ubiquitous transport-level encryption. In Proceedings of the 19th Conference on USENIX Security Symposium, 2010.*

[12] *Boston Globe. Recording industry withdraws suit. http://www.boston.com/business/articles/2003/ 09/24/recording-industry-withdraws-suit/. April 2012.*

[13] *D. Chaum and E. Van Heyst. Group signatures. In Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'91, pages 257–265, Berlin, Heidelberg, 1991. Springer-Verlag.*

[14] *I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Designing Privacy Enhancing Technologies, pages 46–66. Springer, 2001.*

[15] *R. Clayton. Mobile internet access data retention (not!). http://www.lightbluetouchpaper.org/2010/01/14/mobile-internet-access-data-retention-not/. January 2012.*

[16] *S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification, 1998.*

[17] *S. DiBenedetto, D. Massey, C. Papadopoulos, and P. Walsh. Analyzing the Aftermath of the McColo Shutdown. In Applications and the Internet, 2009. SAINT '09. Ninth Annual International Symposium on, pages 157 –160, July 2009.*

[18] *R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In Proceedings of the 13th Conference on USENIX Security Symposium, page 21. USENIX Association, 2004.*

[19] *N. Doraswamy and D. Harkins. IPSec: the new security standard for the Internet, intranets, and virtual private networks. Prentice Hall, 2003.*

[20] *M. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In Proceedings of the 9th ACM Conference on Computer and Communications Security, pages 193–206. ACM, 2002.*

[21] *S. Garfinkel. Pretty Good Privacy (PGP). In Encyclopedia of Computer Science, pages 1421–1422. John Wiley and Sons Ltd., Chichester, UK.*

[22] *S. Gebert, R. Pries, D. Schlosser, and K. Heck. Internet Access Traffic Measurement and Analysis. In TMA, pages 29–42, 2012.*

[23] *Guardian. Nick Clegg tries to head off Lib Dem revolt over email surveillance plans. http://www.guardian.co.uk/uk/2012/apr/03/ theresa-may-email-surveillance-plans. April 2012.*

[24] *A. Haeberlen, R. Rodrigues, K. Gummadi, and P. Druschel. Pretty good packet authentication. In Proceedings of the Fourth Workshop on Hot Topics in System Dependability (HotDep'08), Dec 2008.*

[25] *India News. Techie jailed due to Airtel mistake. http://twocircles.net/node/25440. April 2012.*

[26] *Information Liberation. IP address typo leads to a false arrest in Kansas. http://www.informationliberation.com/?id=1218. April 2012.*

[27] *I. S. Institute. RFC 793, 1981. Edited by Jon Postel. Available at http://rfc.sunsite.dk/rfc/rfc793.html.*

[28] *V. Jacobson, C. Leres, and S. McCanne. libpcap, Lawrence Berkeley Laboratory, Berkeley, CA. Initial public release June, 1994.*

[29] *K. Jang, S. Han, S. Han, S. Moon, and K. Park. Sslshader: cheap ssl acceleration with commodity processors. In Proceedings of the 8th USENIX conference on Networked systems design and implementation, NSDI'11, pages 1–1, Berkeley, CA, USA, 2011. USENIX Association.*

[30] *T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, A. Ghodsi, P. B. Godfrey, N. McKeown, G. Parulkar, B. Raghavan, J. Rexford, S. Arianfar, and D. Kuptsov. Architecting for innovation. SIGCOMM Comput. Commun. Rev., 41(3):24–36.*

[31] *M. E. Kounavis, X. Kang, K. Grewal, M. Eszenyi, S. Gueron, and D. Durham. Encrypting the internet. In Proceedings of the ACM SIGCOMM 2010 conference, SIGCOMM '10, pages 135–146, New York, NY, USA, 2010. ACM.*

[32] *V. T. Lam, S. Antonatos, P. Akritidis, and K. G. Anagnostakis. Puppetnets: Misusing Web Browsers as a Distributed Attack Infrastructure. In Proceedings of the 13th ACM conference on Computer and Communications Security, CCS '06, pages 221–234, New York, NY, USA, 2006. ACM.*

[33] *X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: Secure and Adoptable Source Authentication. In Proceedings of the 5th NSDI., 2008.*

[34] *X. Liu, X. Yang, D. Wetherall, and T. Anderson. Efficient and Secure Source Authentication With Packet Passports. In Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet (SRUTI), pages 2–2, Berkeley, CA, USA, 2006. USENIX Association.*

[35] *S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan. Tcp fast open. In Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies, CoNEXT '11, pages 21:1–21:12, New York, NY, USA, 2011. ACM.*

[36] *M. Reiter and A. Rubin. Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security (TISSEC), 1(1):66–92, 1998.*

[37] *R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM, 26:96–99, January 1983.*

[38] *D. J. D. J. Ryan and G. Shpantzer. Legal aspects of digital forensics. Proceedings Forensics Workshop, (Il), 2002.*

[39] *B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. A1bazaar, 2007.*

[40] *S. Stamm, B. Sterne, and G. Markham. Reining in the web with Content Security Policy. In Proceedings of the 19th International Conference on World Wide Web, WWW '10, pages 921–930, New York, NY, USA, 2010. ACM.*

[41] *S. J. Stolfo, S. M. Bellovin, A. D. Keromytis, S. Hershkop, S. W. Smith, and S. Sinclair, editors. Insider Attack and Cyber Security - Beyond the Hacker, volume 39 of Advances in Information Security. Springer, 2008.*

[42] *P. Syverson. A taxonomy of replay attacks [cryptographic protocols]. In Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings, pages 187–191. IEEE, 1994.*

[43] *TechCrunch. AOL Proudly Releases Massive Amounts of Private Data. http://techcrunch.com/2006/08/06/aol-proudly- releases-massive-amounts-of-user-search-data/.*

[44] *J. Viega, M. Messier, and P. Chandra. Network security with OpenSSL. O'Reilly Media, 2002.*

[45] *A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In IEEE Symposium on Security and Privacy, May 2003.*

[46] *L. Zhuang, F. Zhou, B. Zhao, and A. Rowstron. Cashmere: Resilient Anonymous Routing. In Proceedings of the 2nd NSDI. USENIX Association, 2005.*

[47] *Lobert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. The Click modular router. 17th ACM Symposium on Operating Systems Principles (SOSP '99).*