From protecting protocols to layers: designing, implementing and experimenting with security policies in RINA

Eduard Grasa*, Ondrej Rysavy†, Ondrej Lichtner†, Hamid Asgari‡, John Day§ and Lou Chitkushev§
*Distributed Applications and Networks Area, Fundacio i2CAT, eduard.grasa@i2cat.net
†Faculty of Information Technology, Brno University of Technology, {rysavy, ilichtner}@fit.vutbr.cz
‡Thales Research and Technology, United Kingdom, hamid.asgari@uk.thalesgroup.com
§Computer Science Department, Metropolitan College, Boston University, {day, ltc}@bu.edu

***Abstract*—Current Internet security is complex, expensive and ineffective. The usual argument is that the TCP/IP protocol suite was not designed having security in mind and security mechanisms have been added as add-ons or separate protocols. We argue that fundamental limitations in the Internet architecture are a major factor contributing to the insecurity of the Net. In this paper we explore the security properties of the Recursive InterNetwork Architecture, analyzing the principles that make RINA networks inherently more secure than TCP/IP-based ones. We perform the specification, implementation and experimental evaluation of the first authentication and SDU protection policies for RINA networks. RINA's approach to securing layers instead of protocols increases the security of networks, while reducing the complexity and cost of providing security.**

I. Introduction and motivation

Current Internet security is complex, expensive and ineffective. Attacks on the applications using the Internet and on the infrastructure providing it grow every year, in spite of the ever-increasing security-related protocols, systems and devices deployed in the net. One of the usual explanations is that the TCP/IP protocol suite was not designed having security in mind. Security mechanisms have been added as add-ons or separate protocols. However, two decades after recognizing this issue [1], Internet security levels are still not adequate for a critical infrastructure. We argue that one of the main causes of the Internet's security problems are the fundamental flaws and limitations in the TCP/IP protocol suite design [2], which have caused a proliferation of protocols that try to address some of the issues, making the overall system more complex and difficult to protect.

In this paper we explore the security properties of RINA, the Recursive InterNetwork Architecture [3]. RINA is a fundamental theory of computer networks that sees all networking as distributed Inter Process Communication (IPC). RINA is structured around a single type of layer - called Distributed IPC Facility or DIF - that repeats as many times as needed by the network designer (see Figures 1 and 2). In RINA all layers provide the same service (communication flows between distributed applications) and have the same internal structure. The instantiation of a layer in a computing system is an application process called IPC Process (IPCP). All IPCPs have the same functions, divided into data transfer (delimiting, addressing, relaying, multiplexing, lifetime termination, error check, encryption), data transfer control (flow and retransmission control) and layer management (enrollment, routing, flow allocation, namespace management, resource allocation, security management). The functions of an IPCP are programmable via policies, so that each DIF can adapt to its operational environment and to different application requirements.

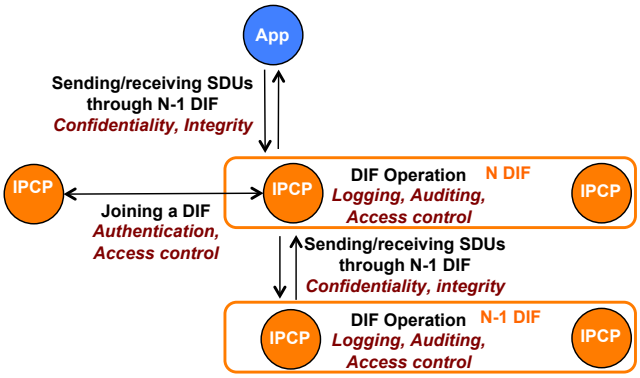

Fig. 1. Placement of security functions in RINA

The rest of the paper is structured as follows. In section II we discuss the RINA design principles that are relevant to security, and compare them to the TCP/IP protocol suite structure, focusing on the security implications of both approaches. Section III describes our main contribution: the specification, design, implementation and experimental evaluation of the first authentication and SDU Protection policies for RINA. Section IV provides concluding remarks and discusses future work.

II. Aspects of RINA design contributing to security

The following paragraphs describe the more important features present in the RINA design that make RINA networks inherently more secure than networks built using the TCP/IP protocol suite - such as the current Internet.

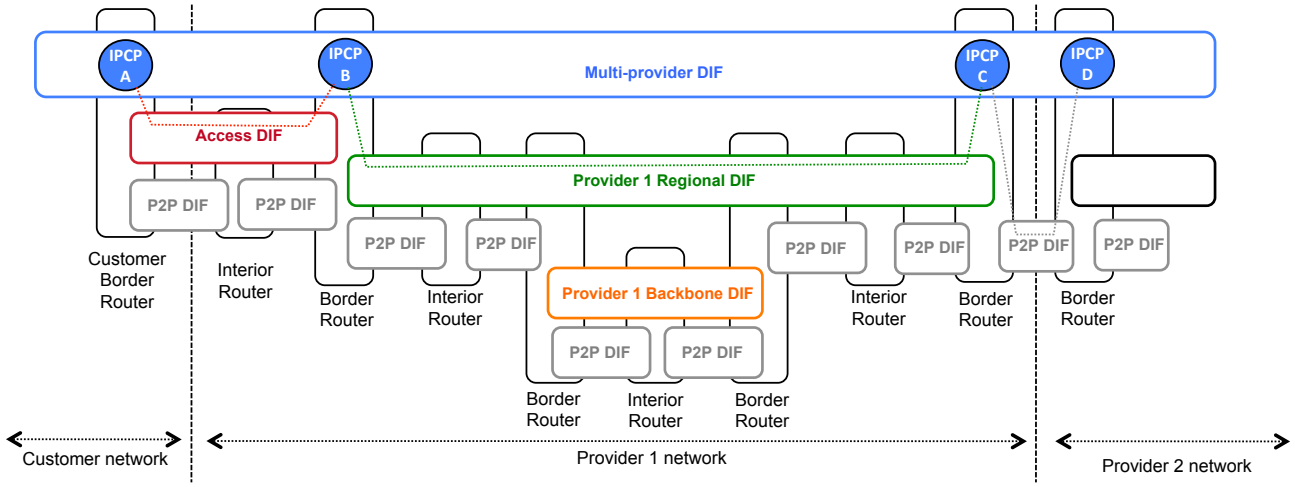

Fig. 2. Using RINAs recursive structure to hide the internal layers of a service providers network. Only provider 1 knows about the green and orange DIFs

A. Secure layers instead of protocols

Layers are the composable building block of RINA, the tool available to network designers to build networks. Layers, and not individual protocols, are the item to be secured in the RINA architecture. The recursive structure of RINA enables a clear security model in which the trust relationships between layers and between the members of a single layer are well understood. Figure 1 illustrates these trust boundaries, which facilitate the placement of the different security functions.

Users of a DIF need to have little trust of the DIF they are using: only that the DIF will attempt to deliver Service Data Units (SDUs) to some process. Applications using a DIF are ultimately responsible for ensuring the *confidentiality* and *integrity* of the SDUs they pass to the DIF. Therefore, proper SDU protection mechanisms (such as encryption) have to be put in place. When a new IPCP wants to join a DIF it first needs to allocate a flow to another IPCP that is a already a DIF member via an N-1 DIF both processes must share in common. Here *access control* is used to determine if the requesting application is allowed to talk to the requested application. If the flow to the existing member is accepted, the next step is to go through an *authentication* phase, the strength of which can range from no authentication to cryptographic schemes. In case of a successful authentication the DIF member will decide whether the new IPCP is admitted to the DIF, executing a specific *access control* policy.

Remote operations on peer IPCPs are another area where *access control* is of key importance. All the layer management functions of an IPCP use a common infrastructure to exchange information with its peers: the Resource Information Base (RIB) and CDAP. CDAP defines a protocol to perform six remote operations over a set of distributed objects, which are used to model the information of each specific layer management task. The RIB imposes a schema (naming and set of relationships) on the DIF objects. At the finest granularity it is possible to take an access control decision to authorize the access to each individual object in the RIB schema for each of the CDAP operations (allowing different tasks to define their own access control restrictions).

Small et al. perform a threat analysis of RINA at the architecture level in [4], concluding that when proper authentication, SDU protection and access control policies are put in place, a DIF is a securable container: a structure used to hold or transport data that can be made not subject to threat. In contrast the TCP/IP protocol suite security model is usually based on building security functions for each protocol. For example, DNSSEC [5] provides data integrity and authentication to security-aware resolvers. IPsec [6] is a general framework for secure IP communications, supporting confidentiality, integrity, authentication or protection against replay attacks. However since IPsec works end-to-end within an IP layer, it either only protects the IP payload (transport mode) or makes IP connection-oriented (tunnel mode), encapsulating a protected IP packet into an unprotected IP packet. This makes IPsec a partial solution, not addressing the requirements of IP control plane protocols, which need to define their own security functions, such as OSPF [7] or BGP [8]. TLS, the Transport Layer Security Protocol [9], specifies a set of related security functions to enable secure communications over the transport layer. All in all, this approach results in more overhead and complexity compared to securing layers. In [10] Small performs an initial comparison between RINA and the current Internet, measuring the flows, protocols and mechanisms required to secure each architecture. Small concludes that RINA networks can deliver on security requirements with less complexity than is currently possible using the Internet protocol suite.

B. Recursion allows for isolation and layers of smaller scope

One of the main challenges in securing the current Internet is that of its scope. The attack surface of a layer increases with its size: the bigger a layer is the larger will be the number of potential attackers and of exploitable vulnerabilities such as missconfigurations, use of weak credentials, etc. The scope of the public Internets IP layer is gigantic; big enough to exhaust IPv4s 32 bits of address space, and it will continue growing

with IPv6. Not only does this fact make nodes in the public Internet more prone to attacks, but also thwarts the deployment of new protocols due to two reasons: either the new protocols need to assume some level of trust in their peers that cannot be guaranteed in the wild public net; or millions of standard defense mechanisms such as firewalls need to be updated to consider the particularities of the new protocol, which is hard to do in practice (e.g. SCTP deployment [11]). Existing protocols are also hard to upgrade, even if the upgrades are critical to the security of the Internet such as the case for BGP security extensions [12].

The advent of network virtualization and its large-scale deployment in datacentres (DCs) has provided DC network designers with a tool to create layers of smaller scope that provide enhanced isolation, minimize the impact of security threats and enable the customization of security policies to different user profiles [13]. RINAs recursive structure generalizes network virtualization, allowing network architects to compose layers of arbitrary size and custom policies into a network with a clear security model.

Figure 2 shows an example of the network of a service provider, connected to a customer network (left) and peering with another providers network (right). Provider 1 only shares the *access DIF* and the *multi-provider* DIF with other networks, the internal layers of the provider - *regional* and *backbone* DIFs - are not visible outside of the providers network. This design reduces the networks attack surface, limiting the damage that an external attacker can perform: most of the provider's routing and resource allocation functions are executed in the internal DIFs. Compromising those DIFs requires physically compromising the providers assets, therefore the provider can focus its resources on protecting the perimeter of its network with strong authentication and SDU protection policies.

C. Separation of mechanism from policy

In RINA the principle of separating mechanism from policy [14] is used to separate the fixed parts of an IPC Process function - which are the same across DIFs - from the variable parts. For example, an acknowledgement is a mechanism, when to acknowledge is policy. This principle enables the same mechanisms to be re-used across DIFs, minimizing the number of different mechanisms present in the network [10] while still allowing for the customization of the DIF security policies. Policies written for a DIF can be re-used in other DIFs, maximizing the efficiency of specifications and implementations.

Separating mechanism from policy allows each DIF to adapt to different operating environments while keeping an upper bound to the complexity of the architecture, which is one of the critical metrics when securing a distributed system [15]. Network architects don't need to design more protocols, just policies that are tailored to the requirements of different DIFs. For example, in Figure 2 *IPCP B* in the *blue DIF* use some form of cryptographic authentication and encryption when exchanging information with *IPCP A* over the *red DIF*, since

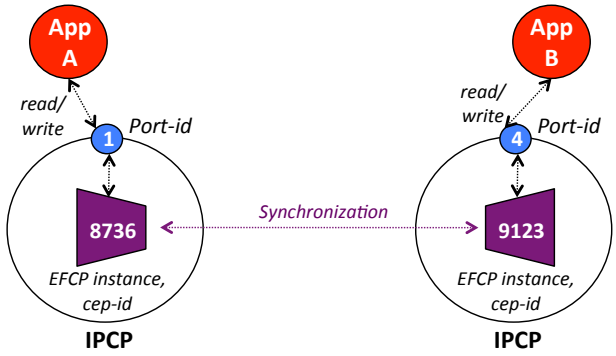

Fig. 3. Decoupling of port allocation and synchronization in RINA

the *red DIF* is shared between the provider and different customers. However *IPCP B* may use no authentication nor encryption when exchanging information with *IPCP C* over the *green DIF*, since in this case all systems are under the control of the provider.

D. Decoupling of port allocation form synchronization

In the TCP/IP Protocol suite TCP overloads the port-id to be both a local handle (socket) and the connection-endpoint-id (cep-id). Furthermore the lack of application names overloads port-ids with application semantics: application endpoints are identified by a combination of IP address and a well-known port-id that is assigned when the application binds to an IP layer. Static destination port-id values have to be known by the source application when requesting a transport connection. Therefore an attacker wanting to intercept a particular TCP connection only needs to guess/spoof the source port-id.

In RINA port-allocation and synchronization are separate functions, applying the results obtained by Watson with the *delta-t* protocol design [16]. The port-allocation procedure is explicitly triggered by an application requesting a flow to a destination application. The source IPCP dynamically assigns a local port-id to the flow and creates an instance of the EFCP protocol that takes care of the feedback synchronization aspects (flow and retransmission control). The EFCP instance is identified by a dynamically generated source cep-id that is mapped to the port-id via a *local binding* as shown in Figure 3. The destination IPCP does the equivalent steps, resulting in a local port-id and a destination cep-id. The source and destination cep-id are the values seen on PDUs in the wire; port-ids are just of local significance and used by applications to read/write data from flows.

The state of ports and connections is managed with different approaches: port state is explicitly created and removed by applications (hard-state) whereas connection state is created and removed following a timer-based approach (soft-state): after long periods of no traffic the connection state is removed, and created again when new traffic is sent/received on the connection. Bodappati et al. showed in [17] how RINA leverages this design to achieve a greater resiliency than TCP/IP to transport-level attacks such as port scanning, connection opening or data transfer.

E. Use of a complete naming and addressing architecture

Naming and addressing design considerations also have a profound impact in the security of network architectures. Since the TCP/IP protocol suite doesn't have application names (DNS is an external directory), IP layers expose addresses to applications. This disclosure of information facilitates spoofing of IP addresses, and in combination with the use of common monitoring tools such as *traceroute* or *ping* allows attackers on end hosts to learn about the addresses of potential targets in a layer - routers or other hosts - as well as the network connectivity graph. Attackers can use this information to setup DDoS attacks by automating the discovery and infection of vulnerable machines [18], or to attack the network infrastructure by gaining control over routers. RINA features a complete network and addressing architecture, with application names and per-layer directories that perform application to IPC Process address resolution. When an application requests a DIF to allocate a flow to a destination application, it provides the source and destination application process names. The DIF internally resolves the destination application name to the address of the IPC Process where the destination application is registered. Due to the existence of application names a layer's addressing information (address format, valid/active addresses) is not divulged outside of the scope of the layer. An attacker in a host cannot address the IPC Processes of a layer unless it joins the layer it wants to attack, which requires authentication.

III. Specification, Design, Implementation and Experimental Validation of Security Policies

In this section we present the specification, design and implementation of authentication and SDU protection policies for RINA. The policies have been developed for the IRATI open source RINA implementation [19], programmed via the Software Development Kit (SDK) designed by the FP7 PRISTINE project. The SDK allows the policies to be plugged into any DIF, maximizing the re-use of the code. We have experimentally evaluated the behaviour of the policies using a simple scenario that reproduces the relevant characteristics of the service provider network depicted in Figure 2.

A. Authentication and SDU Protection policies

Authentication in RINA is part of the Common Application Connection Establishment Phase (CACEP), which two IPC Processes use to establish an application connection. Such application connection enables the IPC Processes to negotiate the id and version of the object set to be exchanged via CDAP, as well as its concrete encoding. Multiple authentication policies can be plugged into CACEP, ranging from no authentication to those using strong cryptographic techniques. We focus on an authentication policy that uses asymmetric keys for mutual authentication of IPCPs, adapting the SSH2 [20] authentication protocol to the DIF environment.

The policy has two differentiated phases, as illustrated in Figure 4. In the first phase both parties securely negotiate a shared secret via the Diffie-Hellman (DH) key exchange

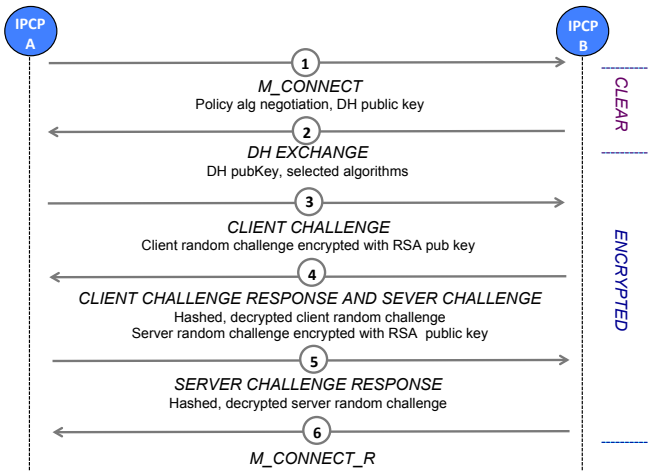

Fig. 4. Message exchanges of the SSH2-based authentication policy

method. The shared secret is then used to generate an encryption key to encrypt all communications between both parties. During the second phase both parties use RSA to authenticate its peer. The policy assumes that both parties use the same RSA key pair, but it can be easily extended to use different sets of keys. Authentication works by an IPCP generating a random byte array, signing it with the RSA public key and sending it to the other IPCP in a challenge message. The receiving IPCP decrypts the challenge with the RSA private key and XORs the result with the shared secret generated via the DH key exchange. The result is hashed via the MD5 algorithm and sent back to the IPCP that issued the challenge, who will perform the equivalent operations to check if the result was correct. The complete specification of this authentication policy can be found at [21], more policies will be specified in the future.

SDU protection is the set of functions used to protect the integrity and confidentiality of traffic when passed on to an underlying DIF. Encryption, error/integrity check, packet lifetime limiting and compression are functions that belong to SDU Protection, but we will just focus on encryption in the context of this paper. The passed SDU is protected according to the policy associated with the underlying DIF. As SDU Protection is fully specified as a policy, there is a considerable flexibility of a degree of security level applied. It is possible to apply a null policy, which does not apply any protective method to the SDU if the underlying DIF is fully trusted. On the other hand, a strong policy involving a robust encryption algorithm can be applied when demanded. In general each layer, being it a DIF or an application layer, should protect its own data from the N-1 layers if they are not trusted. Therefore a default encryption policy for a DIF could just encrypt the traffic generated by the DIF: the headers of data transfer packets, complete data transfer control packets (acknowledgements, etc) and layer management packets. However there may be cases in which the application cannot perform this task and may delegate encryption to a *"VPN DIF"* under the applications control.

In order to support applications that cannot encrypt, we have initially specified a SDU Protection policy that encrypts the

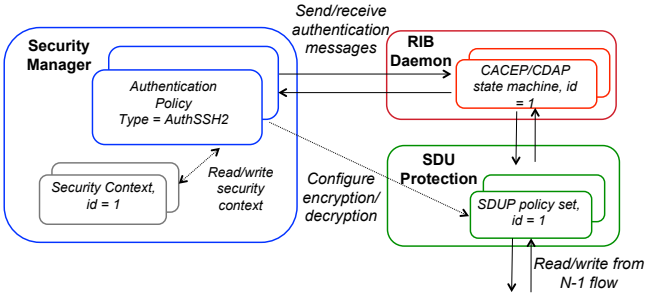

Fig. 5. IPCP components that interact with the policy-set implementation

header and payload of all the packets. The policy does not assume that the N-1 DIF provides flows with reliable and in-order-delivery of SDUs, therefore stream ciphers cannot be employed. Instead, the counter mode is a viable alternative, which has been already developed for IPsec [22]. In counter mode each SDU can be independently encrypted and decrypted from others, so that SDUs can be lost or arrive not in order. It works by combining a nonce and a counter value that increments for each SDU that is encrypted. The combined value is encrypted via a block cipher. The output is XORed with plaintext blocks, and the value of the counter is prepended to the encrypted SDU. Finally a HMAC code is added to the SDU by computing a hash function, in order to protect the integrity of the resulting SDU. Regarding specific algorithms, the Advanced Encryption Standard (AES) with 128 bit keys has been chosen as a block cipher that provides a good computational cost security level tradeoff, while SHA256 has been chosen for the hash function.

B. Design and implementation using the IRATI prototype

IRATI is a C/C++ open source RINA implementation for the Linux/OS. The implementation is divided in three parts: user-space daemons, libraries and kernel components. The IPC Process functions have been split between the user and kernel spaces: the layer management parts are implemented as a user-space daemon, while the data transfer and data transfer control functions reside in the kernel. The IPC Manager Daemon manages the lifetime of IPC Processes in the system, acts as a broker between applications and DIF instances (IPC Processes) and hosts the Management Agent. *Librina* is the library that abstracts out the communication mechanisms between user-space daemons (Netlink sockets) as well as between user-space daemons and the kernel (Netlink sockets and system calls). A more detailed explanation can be found at [19]. The PRISTINE project has developed an SDK to facilitate the programmability of the different components of the implementation, allowing developers to write policies by implementing a set of well-defined C and C++ APIs [23].

Authentication policies are located in *librina*, since they can be used both by the daemons (IPCP, IPC Manager) and applications, within the Security Manager component as illustrated in Figure 5. The Security Manager hosts all the authentication instances supported by the IPC Process, as well as the security contexts of all the N-1 flows currently allocated (the security context stores the configuration of the authentication and SDU protection policies used for a particular N-1 flow). Each authentication policy is in charge of initializing invididual security profiles with the relevant data (algorithms, key material, protection policies).

Authentication policies interact with the RIB Daemon to send/receive messages to its peer, and with *librina* APIs to configure the SDU protection functions in the kernel with the appropriate keys. The RIB Daemon hosts the CACEP state machines, responsible for maintaining the state of the application connection. Each CACEP state machine is associated to an N-1 flow. When a CACEP state machine is in the ”authenticating” state, the RIB Daemon forwards all the messages from the remote peer to the appropriate authentication policy code. The RIB Daemon is also in charge of writing the authentication messages from the authentication policy to the right N-1 flow.

Since the authentication policy needs to perform a number of cryptographic operations, it needs to rely on a proven implementation of those. The openSSL *libcrypto* library has been chosen as a provider of crypto functions for the user-space IRATI daemons, due to its widespread use and extensive feature set. In particular the policy uses DH key and shared secret generation, MD5 and SHA-256 hash functions, RSA key management, public encryption and private decryption.

SDU Protection is designed as a separate module in the kernel part of the IRATI prototype, where it connects to the RMT (Relaying and Multiplexing Task). The SDU Protection module stores the per-port policy configuration, including parameters and state variables negotiated during authentication. After the RMT determines which N-1 port will be used for the SDU transmission it passes the SDU to the SDU Protection module that protects it based on the configuration for this port. In a similar fashion validation of a protected SDU is called from the RMT after it is received from the underlying port.

We have implemented a Proof of Concept (PoC) of the previously described SDU Protection policy in order to test the feasibility of using the native Linux Crypto API for SDU encryption and integration of the basic SDU Protection mechanisms with the rest of the IRATI implementation. Per-port configuration of SDU Protection parameters is stored in the RMT and the N-1 port data structures. The SDU Protection mechanism itself is implemented as part of the serialization and deserialization module, with a series of default policies: TTL as a Lifetime limiting method, 32-bit CRC as an error check method, PKCS#7 padding method and the Linux Crypto API to perform AES128 Electronic Code Book mode encryption. Implementation of cryptographic hashes for message authentication and compression was skipped for the PoC since they can be easily added using the Linux Crypto API.

C. Experimental evaluation

In order to validate the correct operation of the authentication and SDU Protection policies, as well as to demonstrate in practice some of the RINA properties described in section II, we have setup a experiment reproducing the parts of the network service provider scenario described in Figure 2 that

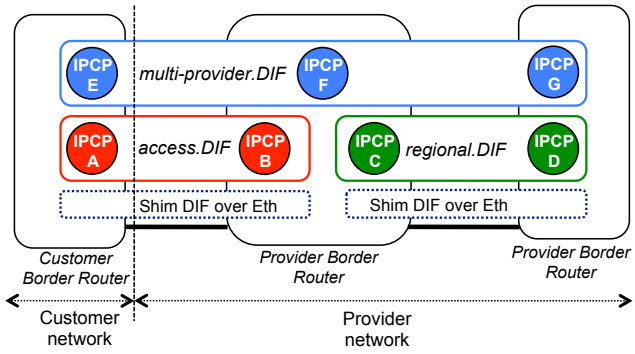

Fig. 6. Experimental scenario based on a simplified service provider

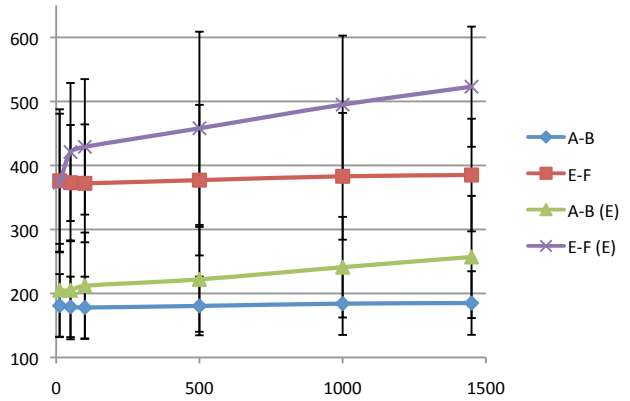

Fig. 7. Application RTT (microseconds) vs. SDU size (bytes) for flows between IPCPs A-B (access DIF) and IPCPs E-F (multi-provider DIF), with and without encryption policies

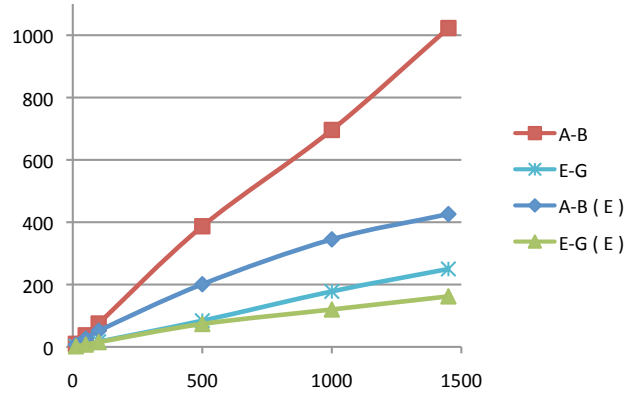

Fig. 8. Application goodput (Mbps) vs. SDU size (bytes) for flows between IPCPs A-B (access DIF) and IPCPs E-F (multi-provider DIF), with and without encryption policies

are most relevant for a security analysis. The goals of the experiment are i) to validate the correct behavior of the policies presented in this paper, further analyzing their impact on the performance of the prototype in terms of goodput and delay; ii) to show a working example of the concepts introduced in section II: policies can be plugged into any DIF, recursion provides isolation, addresses are internal to layers and not exposed to applications using them, security policies protect all the protocols of a given layer.

The experimental scenario is depicted in Figure 6; a tutorial with detailed steps on how to reproduce the experiment is available at [24]. Only the border routers of the provider and customer network are considered. The *access DIF* (in red) allows customers to gain access to higher-level DIFs available through the provider's network. Since the *access DIF* is shared between the provider and multiple customers, cryptographic authentication and SDU protection policies are a good choice to prevent potential attackers to join the DIF and/or intercept the data carried by the DIF. The *regional DIF* (green) is in full control of the provider and not visible to customers or other providers: in this case no authentication and no encryption are feasible policies. The *multi-provider* DIF (blue) is floating on top of the *access* and *regional DIFs*; therefore different policies are advisable for different parts of this DIF. For instance *IPCP E* should use cryptographic authentication and SDU protection policies; *IPCP G* can use no authentication/no encryption while *IPCP F* should use the cryptographic policies for flows over the *access DIF* and can use the no authentication/no encryption approach for flows over the *regional DIF*. The three systems in Figure 6 are XEN VMs hosted in the same physical machine, and use paravirtualized Ethernet drivers for the virtual NICs (not restricted to 1 Gbps as a physical NIC would be).

Figures 7 and 8 show how the RTT and goodput degrade when using the cryptographic SDU protection policies. Without encryption RTT is mostly independent of the SDU size, but the graph shows how encryption introduces a non-negligible processing cost per SDU that causes the RTT to grow with the SDU size. Regarding goodput, performance drops up to 60% for the case of directly connected IPCPs (A-E) when encryption is used. The results for the *multi-provider DIF* (E-G) show less performance degradation for two reasons: encryption is only performed in one hop (over the *access DIF*) and the goodput for the base case (no encryption) was much lower. The big difference in goodput observed between the back-to-back (A-E) and relayed (E-F) experiments can be attributed to the fact that the relaying and scheduling operations are not optimized in the IRATI implementation. Figure 9 shows the time for the enrollment operation to complete when using no authentication and the SSH2-based authentication policy, going from the 10 ms range to approximately 120 ms. Enrollment is the procedure by which an IPCP joins a DIF and acquires enough information to become operational. Since the RTT is very low the cost of performing the policy's cryptographic operations (approx 100 ms) is much higher than the cost of message exchanges (approx 20 ms).

The three DIFs use ovelapping address spaces, which is not a problem since addresses are internal to the layers. When the IPCPs in the *multi-provider* DIF request a flow to the lower DIFs, they just provide the source and destination application names (e.g. *E* to *F*). The SSH-2 based authentication policy and the SDU Protection policies are used in the *multi-provider* and *access* DIFs, securing all the data exchanged between the

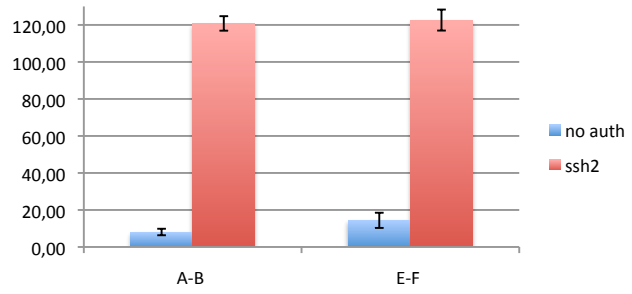

Fig. 9. Enrollment time (ms) between IPCPs A-B (access DIF) and IPCPs E-F (multi-provider DIF), with no authentication (blue) and the SSH2-based authentication policy (red)

IPCPs that use them. IPCP *F* in the *multi-provider* DIF uses different policies depending on the underlying DIFs. All-in-all, the experiment shows a practical example of the flexibility and simplicity of RINA's security model.

IV. Conclusions and Future work

Shortcomings in the structure of the current Internet protocol suite make networks built to the TCP/IP architecture very complex and difficult to secure. The lack of a structured approach towards design causes a proliferation of protocols, each of which has to be individually protected. Unexpected interactions between the ever-increasing protocol-base, inefficiency in the number of repeated security mechanisms, a flat network layer and exposing addresses to applications make the Internet virtually impossible to secure at an affordable cost [25]. In contrast RINA provides an architectural framework with a well-defined building block - the DIF - that recurses as many times as required. Security functions belong to the DIF, not to individual protocols within the DIF. Interactions and the trust model between DIFs are well understood, allowing network architects to reason about the security of a network, understand the threads it is exposed to and design the policies that are more adequate to protect its DIFs.

In this paper we have described the design principles that make RINA networks inherently more secure than current Internet networks, and at a much lower cost. In spite of being an alternative to TCP/IP, RINA networks can be deployed above, below and next to the Internet, facilitating adoption. We have also shown the design, implementation and experimental validation of security policies for DIFs, applying them to a network service provider use case. There is a lot of research and experimentation work to be done in RINA security, we hope that this paper motivates researchers to get involved in this promising approach towards building the networks of tomorrow. Regarding the planned work within FP7 PRISTINE, we will design security policies based on the TLS handshake and record protocols, develop capability-based access control policies for DIFs, investigate credential management approaches and their interaction with network management, and experiment with the results on larger-scale scenarios.

Acknowledgment

This work is partly funded by the European Commission through the FP7 PRISTINE project (Grant 619305). The authors would like to thank all PRISTINE WP4 partners.

References

[1] D. Clark, L. Chapin, V. Cerf, R. Braden, and R. Hobby, "Rfc 1287. towards the future internet architecture," IETF Network Working Group, Tech. Rep., 1991.
[2] J. Day, "How in the heck do you lose a layer!?" in *Network of the Future (NOF), 2011 International Conference on the*, 2011, pp. 135–143.
[3] J. Day, I. Matta, and K. Mattar, ""Networking is IPC": A Guiding Principle to a Better Internet," in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008.
[4] J. Small, "Threat analysis of recursive internetwork architecture difs," Boston University Computer Science Department, Tech. Rep., 2011.
[5] D. Eastlake, "Rfc 2535. domain name system security extensions," IETF, Network Working Group, Tech. Rep., March 1999.
[6] S. Kent and K. Seo, "Rfc 2401. security architecture for the ip protocol," IETF Network Working Group, Tech. Rep., December 2005.
[7] S. Hartman and D. Zhang, "Rfc 6863. analysis of ospf security according to the keying and authentication for routing protocols (karp) design guidelines," IETF Network Working Group, Tech. Rep., March 2013.
[8] M. Lepinski, "Bgpsec protocol sepcification," IETF Network Working Group, Tech. Rep., July 2015.
[9] T. Dierks and E. Rescola, "Rfc 5246. the transport layer security protocol, version 1.2," IETF Network Working Group, Tech. Rep., August 2008.
[10] J. Small, "Patterns in network security: An analysis of architectural complexity in securing rina networks," Boston University Computer Science Department, Master thesis, 2012.
[11] R. Stewart, M. Tuexen, and I. Ruengeler, "Stream control transmission protocol network address translation," IETF Network Working Group, Tech. Rep., June 2011.
[12] R. Lychev, S. Goldberg, and M. Schapira, "Bgp security in partial deployment: Is the juice worth the squeeze?" *Proceedings of ACM SIGCOMM 2013*, pp. 171–182, 2014.
[13] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, 2013.
[14] R. Levin, E. Cohen, W. Corwin, F. Pollak, and W. Wulf, "Policy/mechanism separation in hydra," *Proceedings of the fifth ACM symposium on Operating System principles*, 1975.
[15] B. Schneier, "A plea for simplicity: You can't secure what you don't understand," *Information Security*, 1999.
[16] R. Watson, "Mechanisms for a reliable timer-based protocol," *Computer Networks*, vol. 2, pp. 271–290, 1978.
[17] G. Boddapati, J. Day, I. Matta, and L. Chitkushev, "Assessing the security of a clean-slate internet architecture," *Proceedings of the Network Protocols (ICNP), 2012 20th IEEE International Conference on*, 2012.
[18] J. Mirkovic and P. Reiher, "A taxonomoy of ddos attacks and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.
[19] S. Vrijders, D. Staessens, D. Colle, F. Salvestrini, E. Grasa, M. Tarzan, and L. Bergesio, "Prototyping the Recursive InterNet Architecture: The IRATI project approach," *IEEE Network*, March 2014.
[20] T. Ylonen and C. Lonvick, "Rfc 4252. the secure shell authentication protocol," IETF Network Working Group, Tech. Rep., 2006.
[21] P. consortium, "D4.2. initial specification and proof of concept implementation of innovative security and reliability enablers," FP7 PRISTINE, Tech. Rep., 2015.
[22] R. Housley, "Rfc 3686. using advanced encryption standard (aes) counter mode with ipsec encapsulating security payload (esp)," IETF, Tech. Rep., 2004.
[23] P. consortium, "D2.3. proof of concept rina software development kit," FP7 PRISTINE, Tech. Rep., 2015.
[24] F. PRISTINE, "Security experiments on a small provider network," online: https://github.com/IRATI/stack/wiki/Tutorial-3:-Security-provider-net, October 2015.
[25] B. S. center on International Security, "Risk nexus: Overcome by cyber risks? economic benefits and costs of alternate cyber futures," Zurich Insurance Group, Tech. Rep., 2015.