Firewall Outsourcing: Verifiability and Privacy

Rezwana Reaz
Department of Computer Science
University of Texas at Austin
Austin, Texas, USA
Email: rezwana@cs.utexas.edu

Ehab S. Elmallah
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
Email: elmallah@ualberta.ca

Mohamed G. Gouda
Department of Computer Science
University of Texas at Austin
Austin, Texas, USA
Email: gouda@cs.utexas.edu

Abstract—A firewall system is a packet filter that is placed at the entry point of an enterprise network to examine the packets that attempt to enter the network and decide based on some underlying firewall whether to accept or reject these packets. To simplify the architecture of the firewall system, part of the functionality of the system is outsourced to a public cloud. Unfortunately, public clouds are occasionally unreliable. Therefore, using public clouds to execute part of the functionality of the firewall system makes the firewall system vulnerable to two types of attacks: verifiability attacks and privacy attacks. Prior work in this area yielded outsourced firewall systems that can defend against verifiability attacks or privacy attacks but do not defend against both attacks. In this paper, we present an outsourced firewall system that can defend against these two types of attacks.

Keywords—Firewalls, Firewall System, Outsourcing, Privacy, Verification, Cloud Computing.

I. INTRODUCTION

A firewall system is a packet filter that is placed at the entry point of an enterprise network in the Internet. Packets that attempt to enter the enterprise network through the entry point are examined, one by one, by the firewall system that is placed at the entry point. Examining a packet, the firewall system determines whether to allow the packet to proceed into the enterprise network or to be rejected and prevented from entering the network. Therefore, the function of the firewall system is to identify malicious packets that aim to attack the enterprise network and prevent these packets from entering the network.

Each firewall system has an underlying firewall $F$ so that the firewall system can determine whether to accept or reject an incoming packet according to the rules in the underlying firewall $F$.

A firewall is a sequence of rules where each rule consists of a sequence number, a predicate, and a decision. The sequence number of each rule is a unique integer in the range from 1 to $n$, where $n$ is the number of rules in the firewall. The predicate of each rule is defined using $t$ attributes $u_1$, $u_2$, ..., and $u_t$. The decision of each rule is either “accept” or “reject”.

An example of a firewall $F$ that consists of three rules is as follows.

1. $(u_1 \in [1, 4]) \land (u_2 \in [8, 9])) \rightarrow$ reject
2. $(u_1 \in [2, 4]) \land (u_2 \in [7, 9])) \rightarrow$ accept
3. $(u_1 \in [1, 9]) \land (u_2 \in [1, 9])) \rightarrow$ reject

Note that the predicate of each rule in this firewall $F$ is defined using two attributes $u_1$ and $u_2$ whose integer values are taken from the integer interval $[1, 9]$. The first rule in $F$ is called rule 1, the second rule in $F$ is called rule 2, and so on.

A packet $p$ is defined as a tuple $(b_1, b_2, \ldots, b_t)$ of $t$ integers where each integer $b_i$ is taken from the domain of values of attribute $u_i$.

Now consider two packets $p_1$ and $p_2$ where $p_1$ is defined as the tuple $(u = 3, v = 7)$ and $p_2$ is defined as the tuple $(u = 2, v = 6)$. Packet $p_1$ does not match rule 1, but matches rule 2. So the first match rule in $F$ for packet $p_1$ is rule 2. Similarly, packet $p_2$ does not match rule 1 and rule 2. Rather it matches rule 3. So the first match rule in $F$ for packet $p_2$ is rule 3.

We adopt the notation $\#(F, p)$ to denote the sequence number of the first match rule in firewall $F$ for a packet $p$. For example, the sequence number $\#(F, p_1)$ of the first match rule in $F$ for packet $p_1$ is 2 and the sequence number $\#(F, p_2)$ of the first match rule in $F$ for packet $p_2$ is 3.

For any firewall $F$, we can define a Firewall System that takes as input any packet $p$ and determines whether packet $p$ is accepted or rejected according to the rules in $F$. In this case, we call firewall $F$ the underlying firewall of the firewall system.

As an example, consider a firewall system defined for the underlying firewall $F$ presented above. This system takes as input any packet $p$, for example $(u = 3, v = 4)$, and determines whether $F$ accepts or rejects $p$. Because the first rule in $F$ that matches $p$ is rule 3 and because this rule has a decision “reject”, packet $p$ is rejected by the firewall system.

A firewall system that takes advantage of a public cloud to execute part of the functionality of the firewall system is called an outsourced system. In recent years, with the rise of public cloud computing, enterprises are interested in implementing and managing their firewall systems by taking advantage of a public cloud to reduce the associated cost and management complexity [1]–[3]. According to a survey of 57 firewall systems, firewall outsourcing can provide the following benefits [4]. First, it can reduce the initial investment and the operational cost of the firewall system by taking advantage of the pay-per-use model of the cloud. Second, it can reduce the number or staff needed to manage and implement the firewall system. Third, it can increase availability of the firewall system by maintaining necessary back-ups.
Despite the benefits of firewall outsourcing, an outsourced firewall system is vulnerable to several security attacks caused by the fact that a public cloud is occasionally unreliable. In this paper, we discuss how to design outsourced systems that can defend against attacks that are caused by the fact that public cloud is occasionally unreliable.

II. CATEGORIES OF FIREWALL SYSTEMS

Firewall systems can be classified into two categories: firewall systems without outsourcing, and outsourced firewall systems.

A firewall system without outsourcing refers to a firewall system where the tasks that need to be executed to implement and manage this system are executed by the firewall system without the help of any public cloud.

An outsourced firewall system refers to a firewall system where part of the tasks that need to be executed to implement and manage this system are executed by a public cloud.

In the next two sections, we describe designs of firewall systems without outsourcing and outsourced firewall systems.

III. FIREWALL SYSTEMS WITHOUT OUTSOURCING

In this section, we introduce a design of a firewall system that is built on top of an underlying firewall \( F \) and does not take advantage of any public cloud. As shown in Figure 1, this system consists of two units: a rule matching unit, and a decision unit.

Assume that a packet \( p \) attempts to pass this firewall system. First, packet \( p \) is directed to the rule matching unit of the system. The rule matching unit uses the underlying firewall \( F \) to compute the sequence number \( #(F, p) \) of the first match rule in \( F \) for \( p \). Then, the rule matching unit forwards the pair \((p, #(F, p))\) to the decision unit of the system. Finally, the decision unit takes as input packet \( p \) and the sequence number \( #(F, p) \) received from the rule matching unit and uses firewall \( F \) to compute the decision ("accept" or "reject") of the rule whose sequence number is \( #(F, p) \).

IV. OUTSOURCED FIREWALL SYSTEMS

In this section, we present a design of a firewall system with outsourcing which is shown in Figure 2. The main difference between the system in Figure 1 and the system in Figure 2 is that the rule matching unit in the system in Figure 2 is hosted by a public cloud \( C \). Therefore, the tasks of rule matching unit is outsourced to a cloud \( C \) to obtain the outsourced system in Figure 2.

The outsourced system consists of two units: the rule matching unit which is executed by cloud \( C \), and the decision unit which is executed by the firewall system itself. Both the rule matching unit and the decision unit use the same underlying firewall \( F \).

Assume that a packet \( p \) attempts to pass this outsourced system. First, packet \( p \) is directed to the rule matching unit which is hosted in cloud \( C \). The rule matching unit uses firewall \( F \) to compute the sequence number \( #(F, p) \) of the first match rule in \( F \) for \( p \). Second, the rule matching unit forwards the pair \((p, #(F, p))\) to the decision unit. Third, the decision unit takes as input packet \( p \) and the sequence number \( #(F, p) \) received from the rule matching unit and uses firewall \( F \) to compute the decision ("accept" or "reject") of the rule whose sequence number is \( #(F, p) \).

The decision unit sends the pair \((p, \text{decision of } p)\) to the enterprise network. If the decision of the rule whose sequence number is \( #(F, p) \) is "accept", then in this case the decision for \( p \) is "accept". Otherwise, the decision of the rule whose sequence number is \( #(F, p) \) is "reject" and in this case the decision for \( p \) is "reject".

Correctness of this outsourced system is guaranteed iff cloud \( C \) which hosts the rule matching unit is always reliable, as discussed in Theorem 1 whose proof is in the Appendix.

**Theorem 1:** In the outsourced system in Figure 2 and under the assumption that the cloud in this system is always reliable, the outsourced system takes as input any packet \( p \) and determines whether \( p \) is accepted or rejected according to the underlying firewall \( F \).
On the other hand, if cloud $C$ is occasionally unreliable, then the outsourced system is vulnerable to two types of attacks: verifiability attacks and privacy attacks.

The verifiability attacks, caused by cloud $C$, can be described as follows. When cloud $C$ executes the steps to compute the sequence number $\#(F, p)$ of the first match rule in the underlying firewall $F$ for any incoming packet $p$, $C$ may compute a wrong value $v$. In particular, the computed value $v$ can be the sequence number for a match (but not for a first match) rule in $F$ for $p$.

The privacy attacks, caused by cloud $C$, can be described as follows. If cloud $C$ knows the rules of the underlying firewall $F$, $C$ can leak the underlying firewall $F$ to any potential attacker of the firewall system.

All prior work on designing outsourced firewall systems either defend against verifiability attacks or defend against privacy attacks, but do not defend against both types of attacks. For example, the firewall systems presented in [5] and [6] defend against verifiability attacks, but do not defend against privacy attacks. Also the firewall systems presented in [1], [7]–[11] defend against privacy attacks but not against verifiability attacks.

Our objective in this paper is to design an outsourced firewall system that can defend against both the verifiability and privacy attacks caused by the cloud.

In the following two sections, we present two designs of outsourced systems. The first system is called the verifiable firewall system. This system can defend against verifiability attacks but not against privacy attacks. The second system is called the private system. This system can defend against both verifiability and privacy attacks.

V. VERIFIABLE FIREWALL SYSTEMS

The verifiable system is obtained from the outsourced system presented in Section IV by performing the following three modifications to that system. First, cloud $C$ in the outsourced system is replaced by two identical clouds $C_1$ and $C_2$. We assume that $C_1$ and $C_2$ are occasionally unreliable but they cannot collude. Second, the rule matching unit that is hosted in cloud $C$ is replaced by two identical rule matching units that are hosted in clouds $C_1$ and $C_2$ as shown in Figure 3. Third, the decision unit in the outsourced system is replaced by a verifiable decision unit in the verifiable system. Next, we describe the tasks that need to be performed by the verifiable decision unit.

When a packet $p$ attempts to pass the verifiable system whose underlying firewall is $F$, packet $p$ is first directed to each of the rule matching units hosted in clouds $C_1$ and $C_2$. Each rule matching unit uses firewall $F$ to compute the sequence number $\#(F, p)$ of the first match rule in $F$ for $p$. Let $v_1$ denote the sequence number $\#(F, p)$ computed by the rule matching unit in cloud $C_1$ and let $v_2$ denote the sequence number $\#(F, p)$ computed by the rule matching unit in cloud $C_2$.

The rule matching unit in $C_1$ forwards packet $p$ and the sequence number $v_1$ to the verifiable decision unit. Similarly, the rule matching unit in $C_2$ forwards packet $p$ and the sequence number $v_2$ to the verifiable decision unit.

After the (verifiable) decision unit receives packet $p$ and the two sequence numbers $v_1$ and $v_2$, the decision unit checks whether $v_1$ and $v_2$ are equal. There are two possible outcomes of this check.

First Outcome: The two sequence numbers $v_1$ and $v_2$ are not equal. In this case, the decision unit concludes that the cloud, say $C_1$, that has sent the higher sequence number is lying and the decision for $p$ is “reject”. Second Outcome: The two sequence numbers $v_1$ and $v_2$ are equal. In this case, the decision unit concludes that the two clouds $C_1$ and $C_2$ have sent the same sequence number $v_1$ and that this sequence number is $\#(F, p)$. For this outcome, the decision unit uses the underlying firewall $F$ to compute the decision of the rule whose sequence number is $\#(F, p)$. If the decision of this rule is “accept”, then the decision for $p$ is “accept”. Otherwise, the decision of this rule is “reject” and so the decision for $p$ is “reject”.

After computing the decision for packet $p$, the verifiable decision unit sends the pair $(p, \text{decision of } p)$ to the enterprise network.

Correctness of the first outcome follows from Lemma 1 whose proof is presented in the Appendix.

Lemma 1: Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively, have sent to the verifiable decision unit for the same packet. If $v_1$ and $v_2$ are not equal, then the decision unit concludes (correctly) that the cloud, say $C_1$, that has sent the higher sequence number is lying.

Correctness of the second outcome follows from Lemma 2, discussed below whose proof is in the Appendix, and the fact that the two clouds $C_1$ and $C_2$ are occasionally unreliable, but they can’t collude.

A cloud $C_i$ is occasionally unreliable iff when $C_i$ sends a pair (packet $p$, value $v_i$) to the decision unit, then $v_i$ is the sequence number of any match rule of $p$ that is not necessarily the sequence number of the first match rule of $p$. The two clouds $C_1$ and $C_2$ can’t collude iff when $C_1$ sends the pair $(p, v_1)$ and $C_2$ sends the pair $(p, v_2)$, then $C_1$ and $C_2$ can’t guarantee that $v_1$ equals $v_2$.

Lemma 2: Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively, have sent to the verifiable decision unit for the same packet $p$. If $v_1$ and $v_2$ are equal, then the decision unit concludes (correctly) that the two clouds $C_1$ and $C_2$ have sent the same sequence number $v_1$ and that this sequence number is $\#(F, p)$.

Based on these discussions, correctness of the verifiable system can be stated in Theorem 2 whose proof is in the Appendix.
Theorem 2: Under the assumption that the two clouds in the verifiable systems are occasionally unreliable but they cannot collude, the verifiable system takes as input any packet \( p \) and determines whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

VI. PRIVATE FIREWALL SYSTEMS

The verifiable system that is discussed in the previous section defends against verifiability attacks but is still vulnerable to privacy attacks caused by the fact that cloud \( C_i \) which hosts the rule matching unit of the verifiable system, knows the underlying firewall \( F \). Because \( C_i \) is occasionally unreliable, \( C_i \) can leak the underlying firewall \( F \) to any potential attacker of the system.

To defend against privacy attacks, we design the private firewall system as follows. The private system is obtained from the verifiable system by replacing each rule matching unit that uses firewall \( F \) by a rule matching unit that uses the incomplete version \( IF \) of \( F \).

The incomplete version \( IF \) of \( F \) is the same as \( F \) except that the decisions of all the rules in \( IF \) are unspecified. For example, if the underlying firewall \( F \) is as follows:

1. \( \{ (u_1 \in [1, 4]) \land (u_2 \in [8, 9]) \} \rightarrow \text{reject} \)
2. \( \{ (u_1 \in [2, 4]) \land (u_2 \in [7, 9]) \} \rightarrow \text{accept} \)
3. \( \{ (u_1 \in [1, 9]) \land (u_2 \in [1, 9]) \} \rightarrow \text{reject} \)

then the incomplete version \( IF \) of \( F \) is as follows:

1. \( \{ (u \in [1, 4]) \land (v \in [8, 9]) \} \rightarrow \text{unspecified} \)
2. \( \{ (u \in [2, 4]) \land (v \in [7, 9]) \} \rightarrow \text{unspecified} \)
3. \( \{ (u \in [1, 9]) \land (v \in [1, 9]) \} \rightarrow \text{unspecified} \)

The first rule in this example of \( IF \) is called incomplete rule 1, the second rule in \( IF \) is called incomplete rule 2, and so on. Now consider two packets \( p_1 \) and \( p_2 \) where \( p_1 \) is defined as the tuple \( (u = 3, v = 7) \) and \( p_2 \) is defined as the tuple \( (u = 2, v = 6) \). Packet \( p_1 \) does not match incomplete rule 1, but matches incomplete rule 2. So the first match incomplete rule in \( IF \) for \( p_1 \) is the incomplete rule 2. Similarly, packet \( p_2 \) does not match incomplete rule 1 and incomplete rule 2. Rather it matches incomplete rule 3. So the first match incomplete rule in \( IF \) for \( p_2 \) is the incomplete rule 3.

We adopt the notation \( \#(IF, p) \) to denote the sequence number of the first match incomplete rule in \( IF \) for packet \( p \). For example, the sequence number \( \#(IF, p_1) \) is 2 and the sequence number \( \#(IF, p_2) \) is 3.

Observe that the sequence number \( \#(F, p) \) is equal to the sequence number \( \#(IF, p) \). Therefore, the notations \( \#(IF, p) \) and \( \#(F, p) \) can be used interchangeably.

A packet \( p \) that attempts to pass the private system whose underlying firewall is \( F \), is first directed to each of the rule matching units hosted in clouds \( C_1 \) and \( C_2 \). Each rule matching unit in the private system uses the incomplete firewall \( IF \) instead of the complete firewall \( F \) and computes the sequence number \( \#(IF, p) \) which equals the sequence number \( \#(F, p) \). Each cloud \( C_i \) then sends its computed value \( \#(F, p) \) to the verifiable decision unit along with packet \( p \). The verifiable decision unit of the private system computes the decision for \( p \) in the same way the verifiable decision unit does in the verifiable system. The architecture of the private system is shown in Figure 4. Each occurrence of \( F \) in Figure 3 is replaced by an occurrence of \( IF \) in Figure 4.

![Figure 4. Private firewall system](image)

Note that in the private system, for each incoming packet \( p \), each cloud knows the sequence number \( \#(F, p) \) but does not know the decision of the rule whose sequence number is \( \#(F, p) \). Only the verifiable decision unit in the private system knows this decision. It is possible that each cloud can guess the decision of the rule whose sequence number is \( \#(F, p) \) and can confirm the correctness of this guess by observing the pair \( (p, \text{decision of } p) \) as this pair is transferred from the verifiable decision unit to the enterprise network.

To prevent this possibility, each pair \( (p, \text{decision of } p) \) that is transferred from the decision unit to the enterprise network is encrypted by a secret key that is shared between the verifiable decision unit and the enterprise network. Therefore, neither cloud can confirm the correctness of the guessed pair \((p, \text{decision of } p)\) as this pair is transferred. Therefore, neither cloud can correctly guess the decision of each rule and cannot leak this rule to potential attackers.

Based on these discussions, correctness of the private system is obtained from Theorem 3 whose proof is in the Appendix.

Theorem 3: Each cloud \( C_i \) in the verifiable system knows the underlying firewall \( F \) and can leak \( F \) to potential attackers of the system. By contrast, no cloud \( C_i \) in the private system knows the underlying firewall \( F \) and so cannot leak \( F \) to potential attackers of the system.

VII. RELATED WORK

In the previous two sections, we presented two designs of outsourced systems: the verifiable system and the private system. Each of these systems uses two public clouds that are occasionally unreliable, which makes the system vulnerable to two types of attacks: verifiability attacks and privacy attacks. The verifiable system defends against verifiability attacks but not against privacy attacks. The private system defends against both verifiability attacks and privacy attacks.

Over the past couple of years several outsourced firewall systems [1], [5]–[11] have been presented in the literature. Each of these systems uses one or more public clouds which
can be occasionally unreliable causing the system to be vulnerable to verifiability attacks and privacy attacks. However, each one of these systems either defends against verifiability attacks or defends against privacy attacks. But neither of these systems defends against both attacks. By contrast, the private system presented in this paper defends against both verifiability and privacy attacks.

In the firewall systems presented in [5] and [6], the rules of the underlying firewall \( F \) are stored in the clear in the cloud. Each incoming packet to the enterprise network is directed in the clear to the cloud. For each incoming packet \( p \), the cloud determines whether to accept or reject \( p \) according to the rules of the underlying firewall \( F \) which are stored in the cloud. If the cloud determines to accept \( p \), then the cloud forwards \( p \) to the entry point of the enterprise network. Then the firewall systems in [5] and [6] verify that packet \( p \) is indeed accepted according the underlying firewall \( F \). Therefore, the firewall systems presented in [5] and [6] do not defend against verifiability attacks.

Whereas the firewall system in [6] executes the verification steps online, the firewall system in [5] executes the verification steps offline. Moreover, because the rules of the underlying firewall \( F \) are stored in the clear in the cloud, the cloud can leak these rules to potential attackers of the system. Therefore, the firewall systems presented in [5] and [6] do not defend against privacy attacks.

In the firewall systems presented in [1], [7]–[11], the rules of the underlying firewall \( F \) are encrypted before they are stored in the cloud. Each incoming packet to the enterprise network is directed to the cloud. For each incoming packet \( p \), the cloud determines whether to accept or reject \( p \) according to the encrypted rules of the underlying firewall \( F \) which are stored in the cloud. If the cloud determines to accept \( p \), then the cloud forwards \( p \) to the entry point of the enterprise network. Because the rules of the underlying firewall \( F \) which are stored in the cloud are encrypted, the cloud cannot know the rules of the underlying firewall \( F \) and so cannot leak these rules to potential attackers of the system. Therefore, the firewall systems presented in [1], [7]–[11] do not defend against privacy attacks.

However, none of the firewall systems in [1], [7]–[11] verifies that packet \( p \) that has been forwarded to the entry point of the enterprise network from the cloud is indeed accepted according to the underlying firewall \( F \). Therefore the firewall systems presented in [1], [7]–[11] do not defend against verifiability attacks.

VIII. Concluding Remarks

Our contributions. In this paper, we present a family of firewall systems which is shown in Figure 5. First, we present designs of two categories of firewall systems: firewall systems without outsourcing and outsourced firewall systems. Each of these firewall systems consists a rule matching unit and a decision unit. The firewall system without outsourcing executes the tasks of the rule matching unit and the decision unit without any help from a public cloud. In contrast, the outsourced system outsources the rule matching unit to a public cloud.

Unfortunately, public clouds are occasionally unreliable which makes the outsourced system vulnerable to two types of attacks: verifiability attacks and privacy attacks. To defend against these attacks, we present designs of two outsourced systems: the verifiable system and the private system. The verifiable system outsources the task of the rule matching unit to two public clouds and can defend against verifiability attacks under the assumption that the two public clouds are occasionally unreliable but they cannot collude. However, the verifiable system does not defend against privacy attacks. The private system can defend against both verifiability and privacy attacks.

Another main contribution of this paper is our presentation of the private system which can defend against both verifiability and privacy attacks. Prior work on designing outsourced systems using one or more public clouds either defend against verifiability attacks, for example [5], [6], or defend against privacy attacks, for example [1], [9], [10], but do not defend against both attacks.

Future Works. The private system presented in this paper uses two public clouds and can defend against both verifiability and privacy attacks under the assumption that the two public clouds are occasionally unreliable but they cannot collude. A new extension of the work presented in this paper will be to design outsourced systems that can defend against both verifiability and privacy attacks under the assumption that the public clouds are occasionally unreliable and may collude.

The outsourcing techniques presented in this paper are developed for firewalls. The problem of extending these techniques to outsource other middleboxes in the Internet such as Intrusion Detection Systems (IDS), Network Address Translation (NAT) etc merits further research. Several middlebox outsourcing techniques [6], [8], [11] have been presented in the literature, but neither of these techniques defends against both verifiability and privacy attacks that are caused by outsourcing middleboxes to cloud(s). In this paper, we develop methods to defend against both verifiability and privacy attacks for firewall outsourcing. However, the problems of extending these techniques for outsourcing middleboxes that solve both verifiability and privacy attacks requires further research.

A generalized model of firewalls, called firewall expressions has been discussed in [12]. The problem of extending the outsourcing techniques discussed in this paper to outsource firewall expressions merits further research.

![Figure 5. Our family of firewall systems](image-url)
Statement of Lemma 1: Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively, have sent to the verifiable decision unit for the same packet. If $v_1$ and $v_2$ are not equal, then the decision unit concludes (correctly) that the cloud, say $C_1$, that has sent the higher sequence number $v_1$ is lying.

Proof: Suppose $v_1 > v_2$ and so the decision unit concludes that cloud $C_1$ is lying. We prove, by contradiction, that this conclusion reached by the decision unit is correct. Assume that cloud $C_1$, that has sent the higher sequence number $v_1$ is not lying. This assumption is contradicted by the fact that rules $v_1$ and $v_2$ are also match rules in $F$ for $p$ and the fact that $v_1 > v_2$, which implies that rule $v_1$ is not the first match rule in $F$ for $p$. Therefore, the conclusion, that has been reached by the decision unit that cloud $C_1$ is lying, is correct.

Lemma 2

Statement of Lemma 2: Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively, have sent to the verifiable decision unit for the same packet $p$. If $v_1$ and $v_2$ are equal, then the decision unit concludes (correctly) that the two clouds $C_1$ and $C_2$ have sent the same sequence number $v_1$ and that this sequence number is $\#(F, p)$.

Proof: Suppose the two sequence numbers $v_1$ and $v_2$ are equal for the same packet $p$. We show that the conclusion reached by the decision unit that the two clouds have sent the same sequence number $v_1$ which is equal to the sequence number $\#(F, p)$ is correct.

Recall that the two clouds $C_1$ and $C_2$ are occasionally unreliable, but they can’t collude, so $v_1$ is the sequence number of any match rule of $p$ that is not necessarily the sequence number $\#(F, p)$. Assume that there are $(k+1)$ match rules for packet $p$ in $F$, where $k > 0$. Thus, when the two sequence numbers $v_1$ and $v_2$ are equal, then either the two clouds have sent the sequence number $\#(F, p)$ or they have sent a sequence number of any of the last $k$ match rules for $p$. (Note that when $k = 0$, there exists only one match rule for $p$ in $F$. In this case, if $v_1$ and $v_2$ are equal, then both clouds have sent the sequence number $\#(F, p)$.)

Let $q$ be the probability that the two clouds have sent the sequence number $\#(F, p)$. So the probability that the two clouds have sent a sequence number of any of the last $k$ match rules for $p$ is $(1 − q)$. The following probability analysis shows that $q$ is 1.

Consider the following three predicates, $E$, $G$, and $H$, as follows.

- $E$: $v_1 = v_2$
- $G$: $v_1 = \#(F, p) \land v_2 = \#(F, p)$
- $H$: $v_1 = i_1 \land v_2 = i_2$, where $i_1$ and $i_2$ are randomly selected values from the set of $k$ sequence numbers of the last $k$ match rules in $F$ for $p$.

The probability of the two sequence numbers $v_1$ and $v_2$ are equal is computed as follows.

\[
P(E) = (P(E|G) \times P(G)) + (P(E|H) \times P(H))
\]

where

\[
P(E) = P(v_1 = v_2) = 1,
P(E|G) = P((v_1 = v_2)|v_1 = \#(F, p) \land v_2 = \#(F, p)) = 1,
P(G) = q,
P(E|H) = P((v_1 = v_2)|(v_1 = i_1 \land v_2 = i_2)) = 1/k,
\]

where $i_1$ and $i_2$ are randomly selected values from the set of $k$ sequence numbers of the last $k$ ($> 0$) match rules in $F$ for $p$.

\[
P(H) = (1 − q).
\]

Substituting these probabilities into Equation 1, we conclude that $q$ is 1 for any value of $k$. Therefore, when the two sequence numbers $v_1$ and $v_2$ are equal, the conclusion reached by the decision unit that this sequence number is $\#(F, p)$ is correct.
THEOREM 1

Statement of Theorem 1: In the outsourced system in Figure 2 and under the assumption that the cloud in this system is always reliable, the outsourced system takes as input any packet \( p \) and determines whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

Proof: The outsourced system, shown in Figure 2 consists of a rule matching unit which is executed by cloud \( C \) and a decision unit which is executed by the firewall system. Both the rule matching unit and the decision unit use the same underlying firewall \( F \). We assume that cloud \( C \) is always reliable.

When a packet \( p \) attempts to pass this outsourced system, \( p \) is directed to the rule matching unit which is hosted in cloud \( C \). The rule matching unit uses firewall \( F \) to compute the sequence number \( v \) of the first match rule in \( F \) for \( p \). Since \( C \) is reliable, the computed value \( v \) is indeed the sequence number \( #(F,p) \) of the first match rule in \( F \) for \( p \). After computing \( #(F,p) \), the rule matching unit forwards the pair \( (p, #(F,p)) \) to the decision unit.

The decision unit takes as input packet \( p \) and the sequence number \( #(F,p) \) received from the rule matching unit and uses firewall \( F \) to compute the decision (“accept” or “reject”) of the rule whose sequence number is \( #(F,p) \).

If the decision of the rule whose sequence number is \( #(F,p) \) is “accept”, then in this case the decision for \( p \) is “accept”. Otherwise, the decision for the rule whose sequence number is \( #(F,p) \) is “reject” and in this case the decision for \( p \) is “reject”.

THEOREM 2

Statement of Theorem 2: Under the assumption that the two clouds in the verifiable systems are occasionally unreliable but they cannot collude, the verifiable system takes as input any packet \( p \) and determines whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

Proof: The two clouds \( C_1 \) and \( C_2 \) in the verifiable system are occasionally unreliable. Therefore, when each cloud \( C_i \) sends a pair (packet \( p \), value \( v_i \)) to the decision unit, then \( v_i \) can be the sequence number of any match rule of \( p \) that is not necessarily the sequence number of the first match rule of \( p \).

We proved in Lemma 1 that when the two sequence numbers \( v_1 \) and \( v_2 \) are not equal, then the decision unit of the verifiable system concludes (correctly) that the cloud, say \( C_1 \), has sent the higher sequence number \( v_1 \) is lying.

However, the two clouds \( C_1 \) and \( C_2 \) cannot collude. Therefore, when \( C_1 \) sends the pair \( (p, v_1) \) and \( C_2 \) sends the pair \( (p, v_2) \), then \( C_1 \) and \( C_2 \) can’t guarantee that \( v_1 \) equals \( v_2 \). Using this fact we prove in Lemma 2 that if \( v_1 \) and \( v_2 \) are equal, then the decision unit of the verifiable system concludes (correctly) that the two clouds \( C_1 \) and \( C_2 \) have sent the same sequence number \( v_1 \) and that this sequence number is \( #(F,p) \).

Therefore, when the two clouds \( C_1 \) and \( C_2 \) send the sequence numbers \( v_1 \) and \( v_2 \) to the decision unit, the decision unit verifies whether \( v_1 \) and \( v_2 \) are equal and so they are equal to the sequence number \( #(F,p) \). If \( v_1 \) and \( v_2 \) are equal to \( #(F,p) \), the decision unit determines that the decision for \( p \) is the decision of the rule whose sequence number is \( #(F,p) \). Otherwise, the decision unit determines that one of the two clouds is lying and so the decision for \( p \) is “reject”.

THEOREM 3

Statement of Theorem 3: Each cloud \( C_i \) in the verifiable system knows the underlying firewall \( F \) and can leak \( F \) to potential attackers of the system. By contrast, no cloud \( C_i \) in the private system knows the underlying firewall \( F \) and so cannot leak \( F \) to potential attackers of the system.

Proof:

In the private system, for each incoming packet \( p \), each cloud knows the sequence number \( #(F,p) \) but does not know the decision of the rule whose sequence number is \( #(F,p) \). Only the verifiable decision unit in the private system knows this decision. It is possible that each cloud can guess the decision of the rule whose sequence number is \( #(F,p) \) and can confirm the correctness of this guess by observing the pair \( (p, decision of p) \) as this pair is transferred from the verifiable decision unit to the enterprise network.

To prevent this possibility, each pair \( (p, decision of p) \) that is transferred from the decision unit to the enterprise network is encrypted by a key \( k \) that is shared between the verifiable decision unit and the enterprise network. Therefore, neither cloud can confirm the correctness of the guessed pair \( (p, decision of p) \) as this pair is transferred. Therefore, neither cloud can correctly guess the decision of each rule and cannot leak this rule to potential attackers.