Group Research Report
Protocols, cryptanalysis and mathematical cryptology
Cryptographic Reverse Firewalls

DEJAEGHERE Jules HAMOUCHI Zakariya LALLEMAND Allan
RIVAS Chin THYS Killian

May 17th, 2021

Abstract

The world of IT has never stopped growing since its beginning. The need of security has grown exponentially. In this document, we explain a cryptographic solution (Reverse Firewall) that allows people to communicate securely even when their computer is compromised. First, we identify an attack (Algorithm substitution attack) that occurs on users. Next, we define the notion of reverse firewall and how it can ensure a secure communication even when the user's security has been tampered. We also define the two rounds message protocol used with reverse firewalls that uses the public key of a user and encrypt a plaintext with the chosen public key. Then we explain the notion of hybrid encryption and how it fails with reverse firewalls. We show why the key-agreement process is very important in the context of reverse firewall. We must have a way to exchange keys securely that is why the Diffie-Hellman key exchange is really useful in our case. Furthermore we describe a signature scheme to prove the identity of the two parties. Finally we discuss the limitations of reverse firewalls. One of which is that a powerful attacker that can tamper with the implementation of cryptographic software and hardware could also potentially compromise the reverse firewall.

1 Introduction

In this paper, we show how we can securely exchange messages between parties even when a user's computer is compromised. Nowadays with the rise of computer technology and the multiplication of ways to communicate, it becomes more and more crucial to secure communication against eavesdroppers. In the past few years, we have discovered many vulnerabilities that had not been covered by classical models of cryptography. Indeed, the revelations of Edward Snowden proved that intelligence agencies like the National Security Agency (NSA) of the United States have effectively accessed private information.

First, we present an overview of the related work on the topic in order to understand the motivation behind the reverse firewall framework. We also introduce some principles already in place concerning the reverse firewalls.

Next, we give a definition of the reverse firewall and formal definition related to the properties that a reverse firewall should have. Those definitions allow to have a common understanding for the following sections.

As in [4], we consider the reverse firewall in a public key setting with CPA-secure two-rounds schemes. However, those schemes present a major drawback: they are significantly slower than symmetric key schemes. In order to overcome this limitation, we consider hybrid encryption and show why such schemes are not compatible with reverse firewalls.

As the public key setting is not suitable because of the low speed of execution implied by the public key operations and as the hybrid encryption is not compatible with reverse firewall, we apply reverse firewall to a key agreement process. This key agreement process allows to achieve CCA-security.

Finally, we discuss some structural limitations of the solution proposed in [4] and present some future works done on the topic.

2 State of the art

2.1 Algorithm Substitution Attack (ASA)

In the paper [1], multiples attacks are based on algorithm substitution. In this setup, the attacker is able to subvert the implementation of the users' cryptographic functions. Attackers mounting algorithm substitution attacks are looking for undetectable attacks. In [1] the authors mainly focus on algorithm substitution attacks in the scope of mass surveillance, thus it must hold that the attack will not be detected by the users. Obviously, one of the main requirements for such attacks is that the subverted algorithm maintains correctness of the original scheme. In fact, if the ciphertexts produced by the subverted algorithm are not able to decrypt under the legitimate decryption algorithm the subversion is more likely to be detected soon.

Other properties are desirable (in the point of view of the attacker): the ciphertexts should look like random, even when knowing the legitimate encryption key and the attacker should be able, when looking at the ciphertexts, to learn some information about the plaintext. In the ideal setup, the attacker would be able to recover all the plaintexts exchanged using the subverted algorithm.

In [1], multiple subversions are identified. The first subversion identified takes advantage of the initialization vector in schemes that use randomized initialization vector. The principle of that attack is to use the initialization vector as a way to leak information about the key used to cipher the messages. As the initialization vector is supposed to be random, the subverted algorithm could use an encryption of the private key of the user, under the public key of the attacker, as the initialization vector. If the key size is exactly the size of the initialization vector, the attacker will be able to recover the encryption key in a single time. The same goes if the the key is shorter than the initialization vector, the algorithm have to apply padding before ciphering the key. Finally if the key is longer than the initialization vector, the key must be splitted in several parts and the attacker have to maintain a state to keep track of the different parts of the key already leaked.

The authors discuss about ASA in the context of reverse firewalls and try to demonstrate how a reverse firewall can ensure secure communication even when the implementation of the encryption algorithm is compromised.

2.2 Cryptographic Reverse Firewall (RF)

The main article we are focusing on is a following of Mironov and Stephens-Davidowitz previous work [9]. The RF provides security (confidentiality and integrity) to a user even when his machine has been compromised. The main role of a Reverse Firewall is to protect the communication protocol and its encryption from insider attacks. In order to achieve this protection, the reverse firewall modifies the outgoing and incoming messages from both parties without altering their integrity and by preserving their confidentiality. A reverse firewall cannot be trusted more than the communication channel in use, which means that the RF only has access to public information.

The protection provided by the reverse firewall is based in four principles:

1. Maintain the **functionality** of the underlying communication protocol;
2. The message transmission **security is preserved** even if the computer of one of the parties is compromised by an adversary;
3. The adversaries cannot distinguish between honest and compromised implementation of the message transmission protocol: the RF provides **exfiltration resistance**;
4. Multiples reverse firewalls can be deployed: they are **stackable**.

In [9], Mironov and Stephens-Davidowitz provide the foundation of cryptographic reverse firewalls, limiting their work in an instantiation of private function evaluation and leaving the study of defensive mechanisms against deliberate insider attacks to future works on the subject.

3 Definitions

In simple terms, a reverse firewall W is a stateful algorithm that receives a message and passes that message with a modification. More precisely, for a party A and a reverse firewall W, we define $W \circ A$ as the *composed* party in which W is applied to the messages that the party A receives before really seeing them and also applied to the messages sent by the party A before it leaves *the local network of party A*. Furthermore, W has only access to public parameters of the party A.

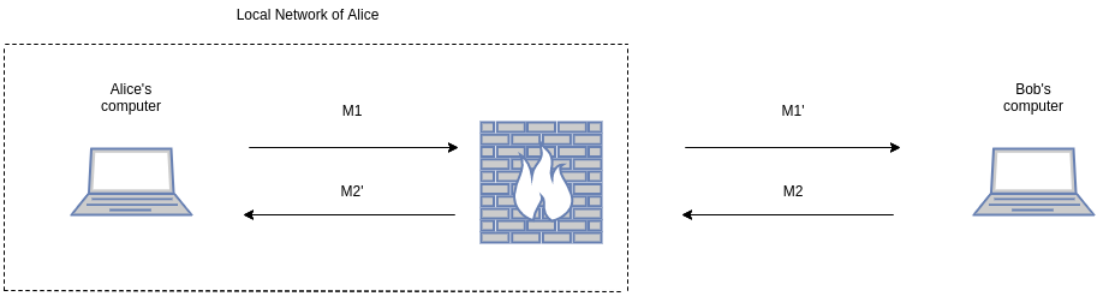

Figure 1: Reverse firewall sits between Alice's computer and Bob's computer, on the local network of Alice

For a better understanding of the following definitions, we will define the terminology. We assume that a cryptographic protocol comes with some functionality F and security requirements S. We use

A to represent the *party* A and $\bar{A}$ to represent the adversarial implementation of A. To represent the functionality-maintaining implementation of A, we will use $\tilde{A}$.

Lastly, we define a protocol P with a party A as such: $P_{A\to\tilde{A}}$. This formula expresses the fact that the role of party A is replaced by party $\tilde{A}$.

In this report, we are interested in firewalls that maintain functionality. For instance, a composed party $W \circ A$ should not break the correctness of the protocol. In addition to this, reverse firewalls should be stackable in series such that $W \circ \cdots \circ W \circ A$ keeps the correctness of the protocol. We define the notion of exfiltration resistance related to the fact that reverse firewalls must be leak resistant. Thus we need a procedure $LEAK$ that show us how it is defined. Indeed, we say that it is exfiltration resistant for Alice and Bob if there is no possibility for Alice to leak information to Bob through the firewall.

proc. $LEAK(\mathcal{P}, A, B, \mathcal{W}, \lambda)$
$(\bar{A}, \bar{B}, I) \leftarrow \varepsilon(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
`IF` $b = 1, A^* \leftarrow \mathcal{W} \circ \bar{A}$
`ELSE`, $A^* \leftarrow \mathcal{W} \circ A$
$\mathcal{T}^* \leftarrow P_{A\to A^*, B\to\bar{B}}(I)$
$b^* \leftarrow \varepsilon(\mathcal{T}^*, S_{\bar{B}})$
`OUTPUT` $(b = b^*)$

Figure 2: LEAK procedure

$LEAK(\mathcal{P}, A, B, \mathcal{W}, \lambda)$ is the security game of exfiltration resistance for a reverse firewall W in protocol P against a party B with input I. ε is the adversary, λ is the security parameter, $S_{\bar{B}}$ is the state of $\bar{B}$ after a run of the protocol P, I is the valid input for protocol P and τ^* is the transcript following from a run of the protocol $P_{A\to A^*, B\to\bar{B}}$ with the input I. Now, let us dive into mathematical definitions regarding our topic.

Functionality preservation, from [4]. A reverse firewall W maintains functionality F for a party A in protocol P if protocol P satisfies F, the protocol $P_{A\to W\circ A}$ satisfies F, and the protocol $P_{A\to W\circ\cdots\circ W\circ A}$ also satisfies F.

The communication should behave as if the the firewalls were totally absent from the network. Both parties shouldn't be able to tell if there is one or more reverse firewalls in the middle of the connection.

Security preservation, from [4]. A reverse firewall preserves security S for a party A in protocol P if protocol P satisfies S, and for any polynomial-time algorithm $\bar{A}$, the protocol $P_{A\to W\circ\bar{A}}$ also satisfies S. The firewall will grant security even when an opponent has compromised A.

The algorithms used in the communication can be used without reverse firewalls and with compromised firewalls. The security level achieved with the reverse firewall will be at most the security of the underlying protocol without any reverse firewall when it is properly implemented. Security is preserved until that all firewalls are compromised and the user is compromised.

Valid transcripts, from [4]. A sequence of bits r and a private input I generate a transcript T in protocol P if a run of the protocol P with input I in which the parties' coin flips are taken from r results in the transcript T. A transcript T is a valid transcript for protocol P if there is a sequence r and private input I generating T such that no party outputs $\perp$ at the end of the run. A protocol has unambiguous transcripts if for any valid transcript T, there is no possible input I and coins r generating T that results in a party outputting $\perp$. Thus a failed run of the protocol should not output a valid transcripts. If it does then it means we have invalid transcripts and the protocol has been broken.

Here, we use the symbol $\perp$ as output for the algorithm when it cannot terminate.

Detectable failure, from [4]. A reverse firewall must detect failure in the sense that it should be able to distinguish between valid and invalid transcripts. This notion will help to identify sub-protocols that have failed. These protocols are part of larger ones that the reverse firewall protects.

A reverse firewall W detects failure for a party A in protocol P if:

- $P_{A \to W \circ A}$ has unambiguous transcripts
- the firewall must output the symbol $\perp$ when running on any invalid transcript for $P_{A \to W \circ A}$
- there is a polynomial-time deterministic algorithm that decides whether a transcript T is valid for $P_{A \to W \circ A}$

In other words, a reverse firewall can tell apart the legitimate messages and tampered messages.

Exfiltration resistance, from [4]. A reverse firewall is exfiltration resistant for a party A using protocol P satisfying functionality F against party B if there is no probabilistic polynomial-time algorithm E with output $\tilde{A}$ and $\tilde{B}$ such that $P_{A \to \tilde{A}}$ and $P_{B \to \tilde{B}}$ satisfy F has a significant advantage in $LEAK(\mathcal{P}, A, B, \mathcal{W}, \lambda)$. If B is empty then we can assume that the firewall is exfiltration resistant.

In other words we say that it is not possible for an adversary to distinguish between an honest implementation of the protocol and a compromised implementation. This prevent him from learning if a user's computer has been compromised.

4 Two rounds message protocol with Reverse Firewall

In [4], the authors consider reverse firewalls in the public key setting with CPA-secure two-round schemes where the first message is the public key of Bob which is randomly chosen over all the possible keys and the second message is the encryption of Alice's plaintext using the public key of Bob.

As mentioned above, the goal of the reverse firewall is to preserve the security of the communication protocol when one of the parties' (Alice and Bob) computer has been compromised. In order to protect the communication protocol from information exfiltration, the RF modifies the encryption of the messages while keeping the functionality of the communication protocol. This will make the legitimate messages and the compromised messages indistinguishable for the adversary.

If, in such setting, we want to apply a reverse firewall on Alice's side, the encryption scheme has to be rerandomizable. Moreover, to apply a reverse firewall on Bob's side, the scheme must be key malleable. The definitions for a rerandomizable and a key malleable scheme are the same as the ones used by the authors of [4]. Intuitively, a scheme is key malleable if a third party, which does not know the private key for the scheme, is able to "rerandomize" a public key and link ciphertexts under that "rerandomized" public key to ciphertexts under the original key.

Before defining what rerandomizable encryption and key malleability are, we first need define the notations used when talking about public-key encryption.

Public-key encryption, from [4]. A public-key scheme is a triple of efficient algorithms $(KeyGen, Enc, Dec)$. $KeyGen$ takes as input 1^λ where λ is the security parameter and outputs a public-key/private-key pair (pk, sk). Enc takes as input the public key and a plaintext m from some plaintext space $\mathcal{M}$ and outputs a ciphertext c from some ciphertext space $\mathcal{C}$. Dec takes as input a ciphertext and the private key and outputs a plaintext or the special symbol $\perp$. We sometimes omit the keys from the input to Enc and Dec and the security parameter input to $KeyGen$. The scheme is correct if $Dec(Enc(m)) = m$ for all $m \in \mathcal{M}$. The scheme is semantically secure if for any adversarially chosen pair of plaintexts (m_0, m_1), $Enc(m_0)$ is computationally indistinguishable from $Enc(m_1)$.

Rerandomizable encryption, from [4]. A semantically secure public-key encryption scheme is rerandomizable if there is an efficient algorithm $Rerand$ (with access to the public key) such that for any ciphertext c such that $Dec(c) \neq \perp$, we have $Rerand(Dec(c)) = Dec(c)$, and the pair $(c, Rerand(c))$

is computationally indistinguishable from $(c, Rerand(Enc(0)))$. We say that it is strongly rerandomizable if the previous property holds even when $Dec(c) = \perp$.

Key malleability, from [4]. A public-key encryption scheme is key malleable if (1) the output of $KeyGen$ is distributed uniformly over the space of valid keys; (2) for each public key pk there is a unique associated private key sk; and (3) there is a pair of efficient algorithms $KeyMaul$ and $CKeyMaul$ that behave as follow. $KeyMaul$ is a randomized algorithm that takes as input a public key pk and returns a new public key pk' whose distribution is uniformly random over the public key space and independent of pk. Let (sk, pk) be a private-key/public-key pair. Let (sk', pk') be the unique pair associated with randomness r such that $pk' = KeyMaul(pk, r)$. Then, $CKeyMaul$ takes as input a ciphertext c and randomness r and returns c' such that $Dec_{sk'}(c) = Dec_{sk}(c')$. We suppress the input r when it is understood. Furthermore, we require that $KeyMaul$ outputs a uniformly random key pk' if called on input that is not in the public-key space.

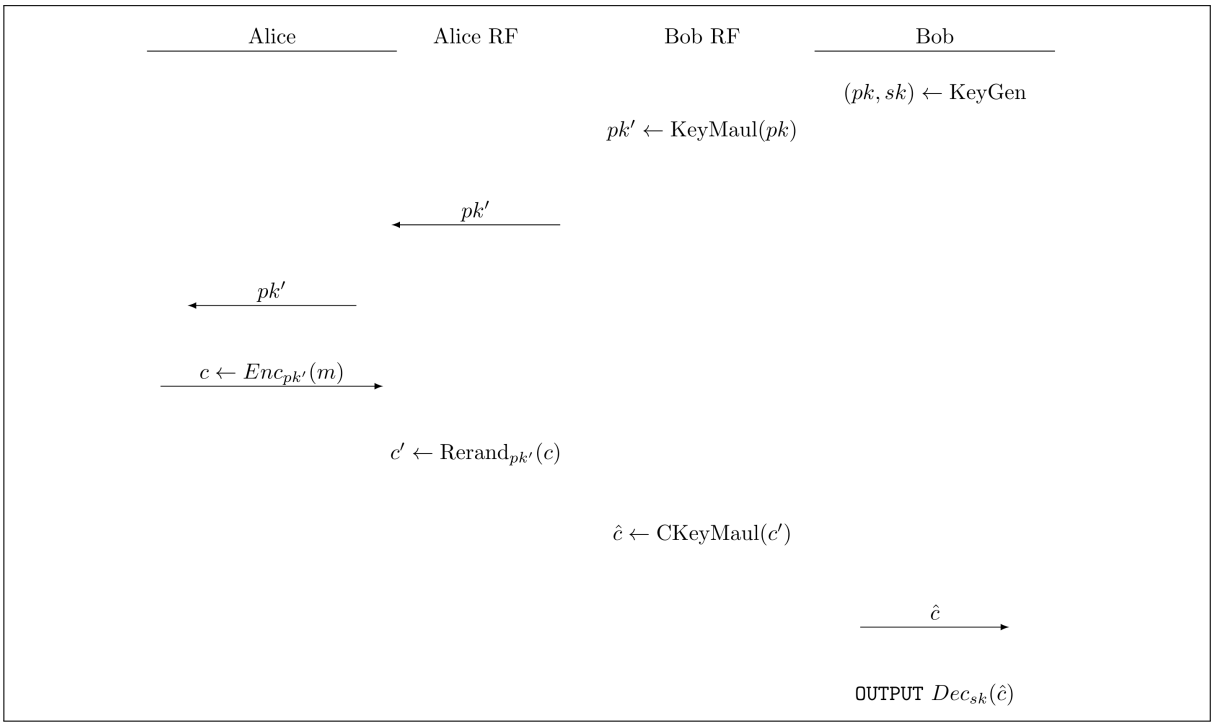

Figure 3: Reverse firewalls on two rounds protocols

While schemes using public key cryptography are simple to implement using reverse firewall, they also have major drawbacks. First, they are only secure against passive adversaries. Second, they are significantly slower than symmetric key protocols when it comes to ciphering and deciphering large amounts of data. Since the plaintexts exchanged by two parties can be quite long, symmetric key cryptography is often preferred over public key cryptography. Public key cryptography is mainly used to agree on a shared secret between two parties who never met before in a secure way.

In the following sections we consider, like the authors of [4], two main solutions to use symmetric key cryptography: hybrid encryption and key agreement. We explain how both solutions perform with cryptographic reverse firewalls and how they could be used in practice to provide better security to the users even when their own computer cannot be trusted.

5 Hybrid encryption

As explained above, it is possible to implement a reverse firewall that maintains functionality, preserves security and resists exfiltration if the underlying encryption scheme is rerandomizable and key malleable. A public-key encryption as ElGamal satisfies both required properties and the desired reverse firewall can be implemented. The widely used solution to improve the encryption protocol efficiency is called hybrid encryption.

5.1 Hybrid encryption scheme

It is called hybrid encryption because this scheme uses asymmetric and symmetric cryptography with the purpose of increase efficiency. In other words, when two parties want to communicate, one of the parties which we would call Alice, generates an uniformly random key (rk) that will be used as secret key in a symmetric key encryption scheme ($SEnc$). Once Alice generated rk, she then uses public-key encryption (pk) to communicate this secret key and a message encrypted with this key ($Enc_{pk}(rk), SEnc_{rk}(m)$) with the other party (Bob). Bob decrypts the secret symmetric key (rk) using his private key and decrypts the symmetric encrypted message in order to validate rk. Since Alice and Bob know rk, they can communicate using symmetric encryption which is a faster scheme than public-key schemes and increases the efficiency of the communication.

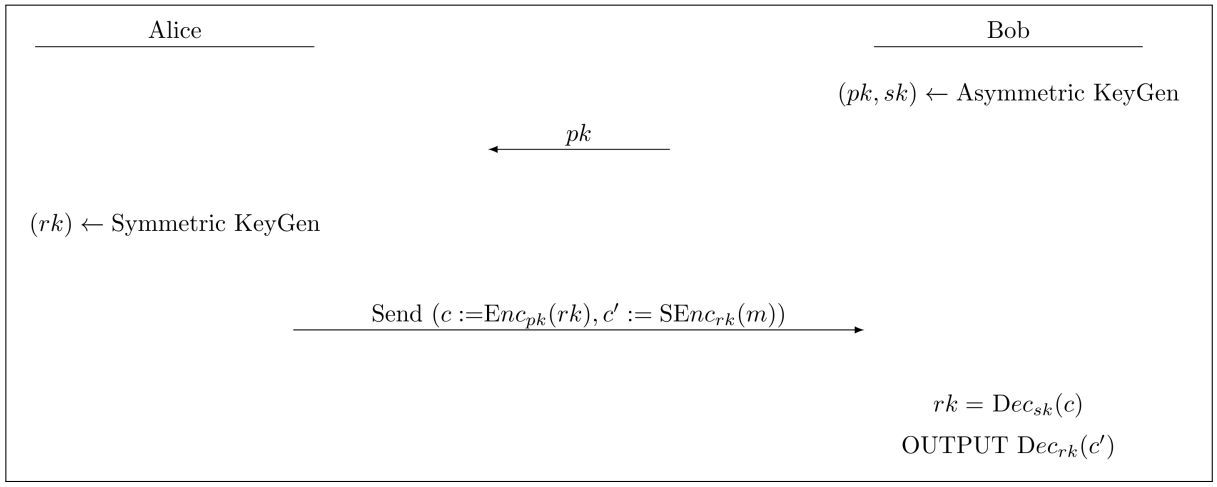

Figure 4: Hybrid encryption scheme

5.2 Hybrid encryption fails with reverse firewall

Hybrid encryption improves efficiency but it fails in preserving the security when the computer, which generate the symmetric key rk, is compromised. By instance, if a tampered version of Alice ($\tilde{A}$) can choose some fixed and valid secret key rk' and send this key to Bob. $\tilde{A}$ maintains functionality but the adversary that knows rk' can read the encrypted messages.

In order to avoid this problem in the context of reverse firewalls, the reverse firewalls must be able to maul the encrypted key $Enc_{pk}(rk)$ into $Enc_{pk}(rk')$ and then convert the plain text encryption $SEnc_{rk}(m)$ into $Senc_{rk'}(m)$. The problem with this, is that it has been proven that any key-malleable symmetric-key encryption scheme implies public-key encryption. **This means that the symmetric-key scheme in the hybrid encryption will require public-key primitives which offers zero advantage in efficiency when comparing to the classic public-key scheme**.

6 Key-Agreement

During the key-agreement process, in the context of reverse firewalls, it is important for the peers to be able to exchange keys in a secure and unalterable way. A widely used solution is the Diffie-Hellman key exchange.

However, this algorithm (and any other algorithm where a peer generates his part of the key after receiving the part of the other peer) implies a problem. Indeed, the second party can generate his part of the key the way he wants.

In the following sections, this problem will be discussed and a solution will be given using reverse firewalls.

Finally, a signature scheme must be added in order to prove the identity of the two peers. Otherwise, the system will not be secure because an attacker could pretend to be one of the two peers.

6.1 The Diffie-Hellman problem

In [4], the authors try to use the Diffie-Hellman key-agreement in order to ensure a secure connection between two parties. A naive construction would be to force both parties to exchange partial keys (g^a for Alice and g^b for Bob in the authors' example) in order to create a shared key g^{ab}. However, this method implies a problem that is related to all protocols where a party can guess the key before influencing it.

To introduce reverse firewall in the key-agreement context, Alice's firewall can rerandomize the public value A (g^a) to obtain a uniformly random message A^α using a single random power $\alpha \in \mathcal{Z}_p$ as shown in figure 5. This reverse firewall maintains the security of the underlying protocol but does not provides a reverse firewall for Bob that maintains correctness and preserve Bob's security. The problem relies in the fact that a compromised computer on Bob's side can have the ability of selectively reject keys and it will be able to re-sample his value b until the key A^{α^b} is formed the way he wants.

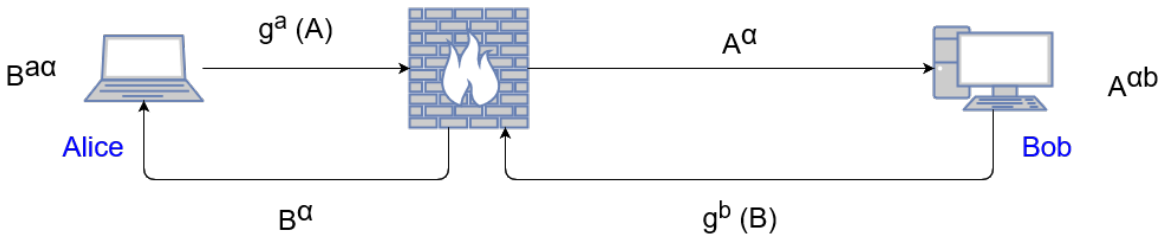

Figure 5: Illustration of the Diffie-Hellman key agreement problem with Alice's reverse firewall

The figure 5 shows the different steps of the Diffie-Hellman key agreement (or any other protocol where Bob could know the first part of the key before generating his part). During the step one, Alice generates her part of the key ($A = g^a$) and sends it. The firewall of Alice chooses a random value α and send A^α to Bob. The problem happens at step two: Bob knows A^α and, in case of a corrupted implementation, can re-randomize b to model the key the way he wants. He now has $A^{\alpha b}$ and sends B (g^b) to Alice. Alice's firewall generates a value α and rerandomizes B to the power α and then sends B^α to Alice. At the last step, Alice generates $B^{a\alpha}$.

6.2 Possible solution using commitment without reverse firewall

Considering this problem, the authors suggest using a three-round protocol to avoid giving any control over the key to the different parties. With this method, Bob creates a commitment of g^b that he will send as first message to Alice. In other words, Alice now has Bob's part of the key but she cannot read it yet. Then, Alice sends her part of the key (g^a) and, afterwards, Bob opens the commitment he made at the beginning of the exchange. The commitment must be rerandomizable to allow the firewalls to rerandomize it and also the whole committed group element.

This approach is a solution to the previous problem: because Bob creates his part of the key and sends its commitment to avoid information disclosure to Alice, otherwise, the problem would remain the same as before. Once Bob has g^a he opens the commitment so Alice can generate g^{ab}. By doing this, it ensures that both parties have to choose their public part of the key before having access or knowing the other party's public part of the key. This will prevent both parties to selectively choose and form the shared key as they want.

6.3 Solution using commitment with reverse firewalls

With reverse firewalls, the commitment scheme has to be key malleable and rerandomizable in order to preserve the security of the protocol. Bob's commitment, let's call it C, is then mauled and rerandomized to give C'. The same rule applies for both parts of the key: the messages are rerandomized and must be malleable. Another reason for the scheme to be rerandomizable is to prevent any leak of information concerning the content of the message during the committing and opening phases.

To ensure the strongest security possible, the scheme has to hide the information using statistical algorithms. Each commitment must also have openings that differs from the previous ones for the same message.

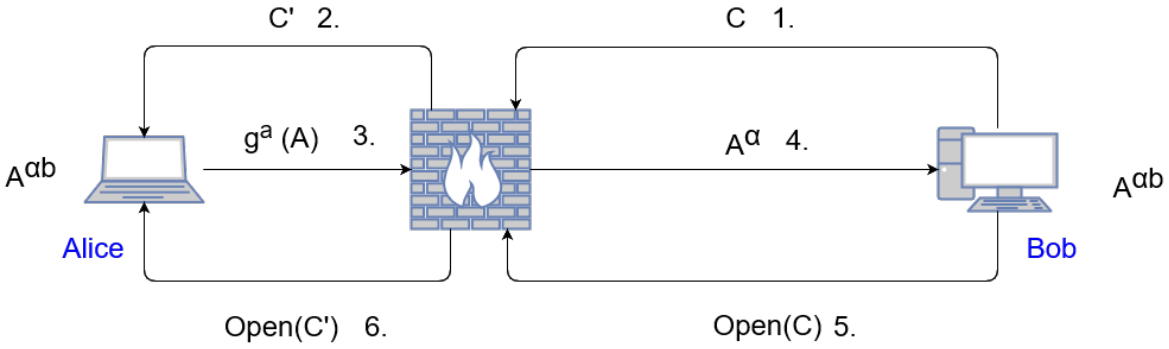

Figure 6: Illustration of the solution using commitments with standard firewalls

A solution would be using a commitment scheme. The process works as follows:

1. Bob sends his commitment for his part of the key. Alice does not know the value commited by Bob.
2. Alice's firewall mauls and rerandomizes the commitment and sends C' to Alice.
3. Alice generates g^a and sends it to Bob.
4. Alice's firewall rerandomizes $g^a = A$ into A^α.

5. Bob opens the commitment.
6. Alice's firewall mauls and rerandomizes the opened commitment.

6.4 Achieve CCA security

In the previous section, there was no possible way to prove the identity of two people wanting to communicate. To overcome this problem, the authors suggest to add a signature and a public key infrastructure and so, achieve IND-CCA security (appendix A). By this way, the attackers will not be able to impersonate one of the parties. The authors suggest different things about the signature:

- As reverse firewalls have to modify the messages, both parties will not see the same transcript. It is therefore not possible to sign the transcript.
- In order for firewalls to maintain security, it is necessary for the reverse firewalls to be able to verify the signature in order to detect invalid signatures.
- The signature should not leak information. The authors suggest using a *unique signature scheme*. In other words, there is only one valid signature for each pair (public key, message). Using this method, the attacker will not be able to rerandomize the signature to model it in a certain way.

The suggested solution is to use a bilinear mapping (appendix B) $e : G \times G \rightarrow G_T$, to sign and to verify the obtained result with both public keys. From Alice and Bob's point of view, they will have to sign the messages with their own private key and verify signatures with the other's party public key. From a reverse firewall's point of view, it will have to verify the signatures using the correct public keys.

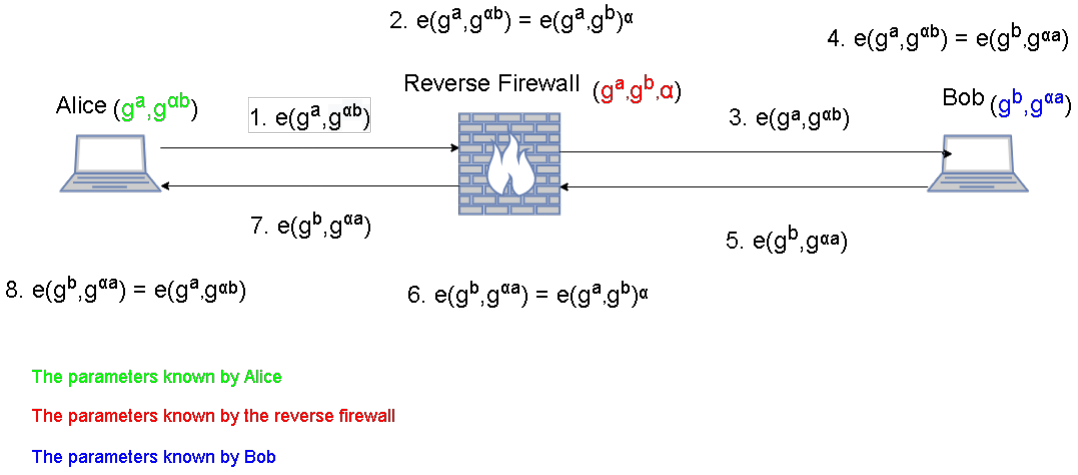

Figure 7: The signing and verification processes in a setup where nothing is compromised. In order to make the scheme easier to read, references to public and private key encryption have been omitted.

1. Alice starts to generate the signature $e(g^a, g^{\alpha b})$ and sends it to the reverse firewall.
2. The reverse firewall generates $e(g^a, g^b)^\alpha$ and compares its value to the value sent by Alice. If both values are equal, then the signature is considered as valid by the reverse firewall.
3. The reverse firewall sends Alice's signature to Bob.
4. Bob generates $e(g^b, g^{a\alpha})$ and compares the obtained value to Alice's one. Once again, the signature is considered valid if the equality is verified.

The steps 5, 6, 7 and 8 correspond to the explanation above for Bob's signature process. If an equality is not obtained at the steps 2, 4, 6 or 8, the signature is considered invalid.

Finally, the authors use a hashing function on the symmetric key obtained from Diffie-Hellman in order to avoid any biases introduced by the bilinear map that could exfiltrate information. This allows to create a symmetric key that is indistinguishable from a random string of bits.

7 Limitations and further works

In this section we identify some limitations present in [4]. Some of these limitations have been studied in future papers in order to find ways to improve the use of reverse firewalls.

A limitation we identified during the review of this paper is the following: the authors of [4] motivate the use of reverse firewalls as a solution to be protected against powerful adversaries that could have the ability to tamper with the implementation of cryptographic software and hardware. However, if such adversaries are powerful enough to tamper with software and hardware it is not impossible to consider that the adversaries will also be able to tamper with reverse firewalls. As explained by the authors, the reverse firewalls preserve security if at least the machine of the person wanting to communicate or the reverse firewalls used by that same person are correctly implemented.

Another limitation is that it is very hard to identify malicious implementations. In fact, it has been shown with popular open-source libraries like OpenSSL and the Heartbleed bug [6]. Even if such popular an well maintained libraries like OpenSSL are subject to critical bugs, it is possible that closed source applications could also be vulnerable. As far as we know, bugs like Heartbleed were not introduced intentionally in the library but powerful attackers may be able to introduce new bugs in open-source software in order to facilitate their attacks.

Limitations mentioned so far are structural limitations and may be out of the scope of the authors' concerns and may require a structural change to the way cryptography is used in software. More practical limitations have been identified and studied by other authors. Some of those limitations are explained in the following paragraphs.

Mironov and Stephens-Davidowitz leave the following open problem for future works "The **Holy grail** would be a full characterization of functionalities and security properties for which reverse firewalls exist." -Mironov and Stephens-Davidowitz [9]. Since then, the functionalities of the reverse firewalls have been studied and applied in different scientific works.

7.1 Cryptographic reverse firewalls for functionalities that are realisable by smooth projective hash proof systems [3]

The authors apply the key malleability and element rerandomizability properties of reverse firewalls, in order to transform the smooth projective hash function (SPHF) into a **malleable SPHF**. In addition to regular SPHFs, malleable SPHFs provide new properties that can be used for the construction of **Generic cryptographic reverse firewalls via malleable SPHFs**. The CRF construction rely on those new properties: *key indistinguishability, element indistinguishability, projection consistency, rerandomozation consistency and membership preservation.* The authors conclude their research by providing a CRF construction for the oblivious transfer protocol.

7.2 Cryptographic reverse firewalls for interactive proof systems [5]

The authors try to construct efficient reverse firewalls for different interactive proof systems in a setting where the implementation of the two honest parties might be corrupted by an adversary. As in the solution studied in [4], the reverse firewalls are transparent: they can be used with existing interactive proof systems without the need to re-implement them. In their paper [5], they define an interactive proof system $\Pi = (P, V)$ is a system that allows a prover P to convince a verifier V that a public statement,

x, is true. The statement x belongs to $\mathcal{L}$, where $\mathcal{L}$ is an NP language and where P and V are modeled as interactive probabilistic polynomial-time machines.

7.3 Reverse firewalls for actively secure multiparty computational protocols [2]

In the article studied in this report[4], the reverse firewalls are used in the context of two-party communication with a passive security preservation. A stronger model is proposed in [2] by implementing a **multiparty computational protocol** in a stronger **actively corruption model** which means that in a multiparty with n parties, an adversary can fully corrupt up to $n-1$ parties and the reverse firewalls can still preserve security. The technique used by the authors to implement such a protocol is based in the novel protocol for multiparty augmented parallel coin-tossing [8].

8 Conclusion

Cryptographic reverse firewalls, if well implemented, can help to protect against strong adversaries that are able to tamper the implementation of the communication protocol or the hardware used for the cryptographic functions. As reviewed above, the reverse firewalls presented in [4] preserve the security of the underlying protocol while maintaining full functionality of the communication. In order to achieve this, the authors introduced the notions of malleability and rerandomization which are used by the reverse firewalls to transform the messages that transit the internet which ensure exfiltration resistance against passive adversaries.

Reverse firewalls can be easily implemented in the context of public-key encryption protocol but the efficiency of the protocol is far from be optimal. Nevertheless, the solution proposed by the authors is to use a Diffie-Hellman key agreement with commitment to securely transmit the public information to create a common key without leaking any information to adversaries that have already compromised the user's computer.

Finally, we outlined some issues not covered by the solution proposed in [4] and we explore other works in order to give an overview of the evolution and the future applications of the reverse firewall framework.

References

[1] Mihir Bellare, Kenneth G Paterson, and Phillip Rogaway. "Security of symmetric encryption against mass surveillance". In: *Annual Cryptology Conference*. Springer. 2014, pp. 1–19.

[2] Suvradip Chakraborty, Stefan Dziembowski, and Jesper Buus Nielsen. "Reverse firewalls for actively secure MPCs". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 732–762.

[3] Rongmao Chen et al. "Cryptographic Reverse Firewall via Malleable Smooth Projective Hash Functions". In: Dec. 2016, pp. 844–876. ISBN: 978-3-662-53886-9. DOI: 10.1007/978-3-662-53887-6_31.

[4] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. "Message transmission with reverse firewalls—secure communication on corrupted machines". In: *Annual International Cryptology Conference*. Springer. 2016, pp. 341–372.

[5] Chaya Ganesh, Bernardo Magri, and Daniele Venturi. "Cryptographic Reverse Firewalls for Interactive Proof Systems". In: *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbr"ucken, Germany (Virtual Conference)*. Ed. by Artur Czumaj, Anuj Dawar, and Emanuela Merelli. Vol. 168. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum f"ur Informatik, 2020, 55:1–55:16. DOI: 10.4230/LIPIcs.ICALP.2020.55. URL: https://doi.org/10.4230/LIPIcs.ICALP.2020.55.

[6] *Heartbleed Bug*. https://heartbleed.com/. Accessed: 2021-04-29.

[7] Bo Hong et al. "Multi-authority non-monotonic KP-ABE with cryptographic reverse firewall". In: *IEEE Access* 7 (2019), pp. 159002–159012.

[8] Yehuda Lindell. "Parallel coin-tossing and constant-round secure two-party computation." In: *Journal of Cryptology* 16.3 (2003).

[9] Ilya Mironov and Noah Stephens-Davidowitz. "Cryptographic reverse firewalls". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 657–686.

[10] Gilles Van Assche and Olivier Markowitch. "Historical ciphers and general principles". In: *Introduction to cryptography (INFO-F-405) - ULB* (2020-2021), pp. 42–43.

A Definition of IND-CPA and IND-CCA security

An encryption scheme is called IND-CPA (Indistinguishability under Chosen Plaintext Attack) if, for two plaintexts p_0 and p_1 chosen by an attacker and the corresponding ciphertexts c_0 and c_1, an attacker is unable to know (with a non negligeable advantage) which ciphertext correspond to which plaintext (the attacker does not know if p_0 correspond to c_0 or c_1) if the attacker does not encrypt the plaintext by himself [10].

An encryption scheme is called IND-CCA (Indistinguishability under Chosen Ciphertext Attack) if it corresponds to the above definition for an attacker that can decrypt the ciphertexts that he wants with exception of c_0 and c_1 [10].

B Definition of Bilinear Groups

We provide the definition of a bilinear group from [7].

Assume G, G_T are multiplicative cyclic groups of a same prime order p, and e is a efficiently computable bilinear mapping $e : G \times G \to G_T$, which satisfies:

1. Bilinearity: For all $g, h \in G, a, b \in \mathbb{Z}_p$, we have $e(g^a, h^b) = e(g, h)^{ab}$;
2. Non-degeneracy: If g_1, g_2 are generators of G respectively, then $e(g_1, g_2) \neq 1$