Message Transmission with Reverse Firewalls—Secure Communication on Corrupted Machines

Yevgeniy Dodis[1], Ilya Mironov[2], and Noah Stephens-Davidowitz[1(✉)]

[1] Department of Computer Science, New York University, New York, USA
noahsd@gmail.com
[2] Google, Menlo Park, USA

Abstract. Suppose Alice wishes to send a message to Bob privately over an untrusted channel. Cryptographers have developed a whole suite of tools to accomplish this task, with a wide variety of notions of security, setup assumptions, and running times. However, almost all prior work on this topic made a seemingly innocent assumption: that Alice has access to a trusted computer with a proper implementation of the protocol. The Snowden revelations show us that, in fact, powerful adversaries can and will corrupt users' machines in order to compromise their security. And, (presumably) accidental vulnerabilities are regularly found in popular cryptographic software, showing that users cannot even trust implementations that were created honestly. This leads to the following (seemingly absurd) question: "Can Alice securely send a message to Bob even if she cannot trust her own computer?!"

Bellare, Paterson, and Rogaway recently studied this question. They show a strong impossibility result that in particular rules out even semantically secure public-key encryption in their model. However, Mironov and Stephens-Davidowitz recently introduced a new framework for solving such problems: reverse firewalls. A secure reverse firewall is a third party that "sits between Alice and the outside world" and modifies her sent and received messages so that *even if the her machine has been corrupted*, Alice's security is still guaranteed. We show how to use reverse firewalls to sidestep the impossibility result of Bellare et al., and we achieve strong security guarantees in this extreme setting.

Indeed, we find a rich structure of solutions that vary in efficiency, security, and setup assumptions, in close analogy with message transmission in the classical setting. Our strongest and most important result shows a protocol that achieves interactive, concurrent CCA-secure message transmission with a reverse firewall—i.e., CCA-secure message transmission on a possibly compromised machine! Surprisingly, this protocol is quite efficient and simple, requiring only four rounds and a small

Y. Dodis—Partially supported by gifts from VMware Labs and Google, and NSF grants 1319051, 1314568, 1065288, 1017471.

N. Stephens-Davidowitz—Partially supported by National Science Foundation under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

© International Association for Cryptologic Research 2016
M. Robshaw and J. Katz (Eds.): CRYPTO 2016, Part I, LNCS 9814, pp. 341–372, 2016.
DOI: 10.1007/978-3-662-53018-4_13

constant number of public-key operations for each party. It could easily be used in practice. Behind this result is a technical composition theorem that shows how key agreement with a sufficiently secure reverse firewall can be used to construct a message-transmission protocol with its own secure reverse firewall.

1 Introduction

We consider perhaps the simplest, most fundamental problem in cryptography: secure message transmission, in which Alice wishes to send a plaintext message to Bob without leaking the plaintext to an eavesdropper. Of course, this problem has a rich history, and it is extremely well-studied with a variety of different setup assumptions and notions of security (e.g., [4]). There are many beautiful solutions, based on symmetric-key encryption, public-key encryption, key agreement, etc.

However, in the past few years, it has become increasingly clear that the real world presents many vulnerabilities that are not captured by the security models of classical cryptography. The revelations of Edward Snowden show that the United States National Security Agency successfully gained access to secret information by extraordinary means, including subverting cryptographic standards [3,34] and intercepting and tampering with hardware on its way to users [23]. Meanwhile, many (apparently accidental) security flaws have been found in widely deployed pieces of cryptographic software, leaving users completely exposed [12–14,25,28]. Due to the complexity of modern cryptographic software, such vulnerabilities are extremely hard to detect in practice, and, ironically, cryptographic modules are often the easiest to attack, as attackers can often use cryptographic mechanisms to mask their activities or opportunistically hide their communications within encrypted traffic. This has led to a new direction for cryptographers (sometimes called "post-Snowden cryptography"), which in our context is summarized by the following (seemingly absurd) question: "How can Alice and Bob possibly communicate securely when an eavesdropper might have corrupted their computers?!"

Motivated by such concerns, Bellare, Paterson, and Rogaway consider the problem of securely encrypting a message when the encrypting party might be compromised [7]. They consider the case in which the corrupted party's behavior is indistinguishable from that of an honest implementation. Even in this setting, their main result shows that even a relatively weak adversary can break any scheme that "non-trivially uses randomness." (They also provide a nice deterministic symmetric-key solution, which we use as a subprotocol in the sequel.) In particular, it is easy to see that a semantically secure public-key message transmission is impossible in their framework. (See [5] for an analysis of weaker notions of security for public-key encryption in this setting.)

1.1 Reverse Firewalls

Due to the strong restriction proved in [7], we consider a relaxation of their model in which we allow for an additional party, a *(cryptographic) reverse firewall* (RF) as recently introduced by Mironov and Stephens-Davidowitz [31]. We provide formal definitions in Sect. 2.1, but since RFs are quite a new concept (and they can be rather confusing at first), we now provide a high-level discussion of some of the salient aspects of the reverse-firewall framework.

A reverse firewall for Alice is an autonomous intermediary that modifies the messages that Alice's machine sends and receives. The hope is that the protocol equipped with a reverse firewall can provide meaningful security guarantees for Alice *even if her own machine is compromised.* As we explain in detail below, the firewall is *untrusted* in the sense that it shares no secrets with Alice, and in general we expect Alice to place no more trust in the firewall than she does in the communication channel itself.

More concretely, Mironov and Stephens-Davidowitz start by considering an arbitrary cryptographic protocol that satisfies some notions of functionality (i.e., correctness) and security.[1] For example, perhaps the simplest non-trivial case is semantically secure message transmission from Alice to Bob, which has the functionality requirement that Bob should receive the correct plaintext message from Alice and the security requirement that a computationally bounded adversary "should not learn anything about Alice's plaintext message" from the transcript of a run of the protocol. Formally, we can model this functionality by providing Alice with an input plaintext and requiring Bob's output to match this, and we can model semantic security by a standard indistinguishability security game.

A reverse firewall for Alice in some protocol *maintains functionality* if the protocol "with the firewall in the middle" achieves the same functionality as the original protocol. E.g., in the case of message transmission, Bob should still receive Alice's message—his output should still match Alice's input. More interestingly, the firewall *preserves security* if the protocol with the firewall is secure even when we replace Alice's computer with some arbitrarily corrupted party. For example, a reverse firewall for Alice preserves semantic security of message transmission if a computationally bounded adversary "learns nothing about Alice's plaintext message" from the transcript of messages sent between the firewall and Bob, *regardless of how Alice behaves.* E.g., the firewall may rerandomize the messages that Alice sends in a way that makes them indistinguishable from random from the adversary's perspective, regardless of Alice's original message. (We analyze such protocols in a stronger setting in Sect. 3, and in a different setting in Appendix B.)

Note that it also makes sense to consider reverse firewalls for the receiver, Bob. For example, consider a protocol in which Bob first sends his public key

[1] The notion of functionality in [31] is quite simple, and it should not be confused with the much more complicated concept of functionality used in the universal composability framework. Formally, Mironov and Stephens-Davidowitz define a functionality requirement as any condition on the output of the parties that may depend on the input, and in practice, these requirements are straightforward.

to Alice, and Alice responds with an encryption of her message under this key. Clearly, if Bob's computer is corrupted in such a protocol, this can compromise security, even if Alice behaves properly. In such a protocol, a firewall for Bob might rerandomize his public key. Of course, to maintain correctness, this firewall must also intercept Bob's incoming messages and convert ciphertexts under this rerandomized key to encryptions under Bob's original key. (Again, see Sect. 3 for a formal treatment of such protocols.)

A key feature of protocols with reverse firewalls, as defined in [31], is that they should be functional and secure *both* with the reverse firewall *and* without it. I.e., there should be a well-defined *underlying protocol* between Alice and Bob that satisfies classical functionality and security requirements. This is one important difference between reverse firewalls and some similar models, such as the mediated model [1] and divertible protocols [9,10,32], and it comes with a number of benefits. ([31] contains a thorough comparison of many different related models.) First, it means that these protocols can be implemented and used without worrying about whether reverse firewalls are present—one protocol works regardless; we simply obtain additional security guarantees with an RF.

Second, and more importantly, this definitional choice provides an elegant solution to a natural concern about reverse firewalls: What happens when the firewall itself is corrupted? Of course, if *both* Alice's own machine and her firewall are compromised, then we cannot possibly hope for security. But, if Alice's own implementation is correct and the firewall has been corrupted, then we can view the firewall as "part of" the adversary in the firewall-free protocol between Alice and Bob. Since this underlying protocol must itself be secure, it trivially remains secure in the presence of a corrupted firewall.[2] This is why we can say that the firewall is trusted no more than the communication channel.

Of course, the advantage of using a firewall comes when Alice's machine is corrupted but the firewall is implemented correctly, in which case the firewall provides Alice with a security guarantee that she could not have had otherwise. In short, *the firewall can only help*. (See Table 1.) In fact, we even require firewalls to be "stackable," so that arbitrarily many firewalls may be deployed, and security is guaranteed as long as *either* (1) Alice's own machine is uncorrupted; or (2) at least one of these firewalls is implemented correctly and honestly.

Finally, it is convenient to identify a class of *functionality-maintaining corruptions*: compromised implementations that are "technically legal" in the sense that they may deviate arbitrarily from the protocol, as long as they do not break its functionality. Some of our reverse firewalls are only secure against this type of corruption. (This model is introduced by [31], and the authors call security against unrestricted compromise *strong* security.) We emphasize that, while this restricted class of compromised implementations is not ideal, it is still quite large. In particular, all of the real-world compromises mentioned above fall into this category [3,12–14,23,25,28,34,40], as do essentially all other forms of compromise considered in prior work, such as backdoored PRNGs [21],

[2] Technically, this statement only holds if the underlying protocol is secure against active adversaries.

Table 1. Security of a protocol with a secure reverse firewall for Alice in various different scenarios.

Alice	Alice's RF	Secure?	
Honest	Honest	✔	Trivial
Honest	Compromised	✔	Underlying protocol's security
Compromised	Honest	✔	RF's security
Compromised	Compromised	✗	Everyone is compromised!

Algorithm Substitution Attacks [7], subliminal channels [39], etc. (We discuss functionality-maintaining corruption in our setting in more detail in the full version [22]. See [31] for a detailed discussion of the general reverse framework.)

1.2 Our Results

In this section, we walk through the results that we obtain in different settings, starting with simpler cases and working our way up to our stronger results. In what follows, Alice is always the sender and Bob is always the receiver of the message. All of our security notions apply to the concurrent setting, in which the adversary may instantiate many runs of the protocol simultaneously. (The proofs are in the full version [22].)

The Symmetric-Key Setting. In the first and simplest scenario, Alice and Bob have a shared secret key. (See Appendix A.) Quite naturally, Alice might want to use a symmetric-key encryption scheme to communicate with Bob. Using a standard scheme (e.g., AES-CBC) would, however, expose her to a number of "algorithm-substitution attacks" (what we call corruption or compromise) described by Bellare, Paterson, and Rogaway [7], such as IV-replacement or a biased-ciphertext attack. To defend against such attacks, Bellare et al. propose using a clever solution: a *deterministic* encryption scheme based on either a counter or a nonce. We briefly consider this case, observing that their solution corresponds to a one-round protocol in our model (in which the firewall simply lets messages pass unaltered).

Unfortunately, we show that strong security (i.e., security against corrupted implementations of Alice that are not necessarily functionality-maintaining) is not achievable without using (less efficient) public-key primitives, even in the reverse-firewalls framework. This provides further motivation to study reverse firewalls in the public-key setting.

Rerandomizable Encryption. As we mentioned earlier, the simplest non-trivial reverse firewall in the public-key setting uses CPA-secure rerandomizable public-key encryption. In particular, Alice can send her plaintext encrypted under Bob's public key, and Alice's reverse firewall can simply rerandomize this ciphertext. We observe that this folklore technique works in our setting. In Sect. 3, we present a generalization of this idea that does not require any public-key infrastructure, by having Bob send his public key as a first message. Following [31], we observe that Bob can have a reverse firewall for such a protocol that rerandomizes his key (and converts Alice's ciphertext from an encryption under the rerandomized key to an encryption under Bob's original key). We therefore show a simple two-round protocol with a reverse firewall for each party.

While such protocols are simple and elegant, they have two major drawbacks. First, they are only secure against passive adversaries (we will return to this issue soon). Second, and arguably more importantly, such protocols require the computation of public-key operations on the entire plaintext. Since plaintexts are often quite long and public-key operations tend to be much slower than symmetric-key operations, it is much faster in practice to use public-key operations to transmit a (relatively short) key for a symmetric-key encryption scheme and then to send the plaintext encrypted under this symmetric key. There are two general methods for transmitting this key in the classical setting: hybrid encryption and key agreement.

Failure of Hybrid Encryption. Unfortunately, hybrid encryption does not buy us anything in the reverse-firewalls framework. Recall that in a hybrid encryption scheme, Alice selects a uniformly random key rk for a symmetric-key scheme and sends rk encrypted under Bob's public key together with the encryption of her message under the symmetric-key scheme with key rk. We might naively hope that we can build a reverse firewall for such a scheme by simply applying the "rerandomizing firewall" to the "public-key part" of Alice's ciphertext. But, this does not work because of the attack in which a corrupted implementation of Alice chooses a "bad key" rk^* with which to encrypt the message. The "bad key" rk^* might be known to an adversary; might be chosen so that the ciphertext takes a specific form that leaks some information; or might otherwise compromise Alice's security. So, intuitively, a reverse firewall in such a scheme must be able to rerandomize the key rk, and it therefore must be able to convert an encryption under some key rk into an encryption of the same plaintext under some new key rk'. Unfortunately, we show that any such "key-malleable" symmetric-key encryption scheme implies public-key encryption. Therefore, it cannot be faster than public-key encryption and is useless for our purposes.

Key Agreement. Recall that a key-agreement protocol allows Alice and Bob to jointly select a secret key over an insecure channel. Security requires that the resulting key is indistinguishable from random to an eavesdropper. Such a protocol is often used in conjunction with symmetric-key encryption in the classical setting, where it is justified by composition theorems relating the security of the message-transmission protocol to the underlying key-agreement protocol. Indeed, we give an analogous result (Theorem 2) that works in our setting, showing that a carefully designed key-agreement protocol with sufficiently secure reverse firewalls can be combined with symmetric-key encryption to produce an efficient CPA-secure message-transmission protocol with secure reverse firewalls.

This motivates the study of key-agreement protocols with secure reverse firewalls. As a first attempt at constructing such an object, we might try to somehow rerandomize the messages in the celebrated Diffie-Hellman key-agreement protocol, in which Alice first sends the message g^a, Bob then sends g^b, and the shared key is g^{ab}. (See Fig. 8.) Here, we run into an immediate problem. Since the firewall must maintain correctness, no matter what message A^* the firewall sends to Bob, it must be the case that the final key is A^{*b}, where b is chosen by Bob. But, this allows a corrupt implementation of Bob to influence the key. For example, Bob can repeatedly resample b until, say, the first bit of the key A^{*b} is zero, thus compromising the security. It is easy to see that this problem is not unique to Diffie-Hellman and in fact applies to *any* protocol in which "a party can learn what the key will be before sending a message that influences the key."

So, to truly prevent any party from having any control over the final key, we use a three-round protocol in which Bob sends a *commitment of* g^b as his first message. Alice then sends g^a, and Bob then opens his commitment.[3] Of course, the commitment scheme that Bob uses must itself be rerandomizable and malleable, so that the firewall can both rerandomize the commitment itself *and* the committed group element. Fortunately, we show that a very simple scheme, a natural variant of the Pedersen commitment, actually suffices.

Since this simple protocol is unauthenticated, it cannot be secure against active adversaries. While passive security might be sufficient in some settings (powerful adversaries are known to passively gather large amounts of web traffic [23]), it would be much better to achieve security against active adversaries. We address this next.

CCA-Security From Key Agreement. We attempt to construct a reverse firewall that preserves CCA-security (i.e., security against active adversaries that may "feed" Alice and Bob arbitrary adversarial messages and read Bob's output). In this setting, we again prove a generic composition theorem, which shows that it suffices to find a key-agreement protocol with a reverse firewall that satisfies certain security properties. In analogy with the passive setting, after agreeing to a key with Bob, Alice can use symmetric-key encryption to send the actual plaintext message. The resulting protocol is CCA-secure, and Alice's reverse firewall preserves this security. (See Theorem 4.)

To instantiate this scheme, we must construct a key-agreement protocol that is secure against active adversaries and has a reverse firewall that preserves this security. Unfortunately, many of the common techniques used in classical key-agreement protocols (or even protocols that are secure against key control) are useless here. In particular, most key-agreement protocols that achieve security

[3] We note that the problem that we face here is very similar to the problem of key control, and our solution is similar to solutions used in the key-control literature. See, e.g., [18].

against active adversaries do so by essentially having both parties sign the transcript at the end of the protocol. Intuitively, this allows the parties to know if the adversary has tampered with any messages, so that they will never agree to a key if a man in the middle has modified their messages. But in our setting, we actually *want* the firewall to be able to modify the parties' messages. We therefore need to somehow find some unique information that the parties can use to confirm that they have agreed to the same key without preventing the firewall from modifying the key. Furthermore, we need the firewall to be able to check these signatures, so that it can block invalid messages. Therefore, our primary technical challenge in this context is to find a protocol with some string that (1) uniquely identifies the key; (2) does not leak the key; (3) respects the firewall's changes to the parties' messages; and (4) is efficiently computable from the transcript. And, of course, the protocol must be secure against active adversaries, even though it is in some sense "designed to help a man in the middle."

In spite of these challenges, we construct a protocol with a reverse firewall for each party that preserves security against active adversaries. Remarkably, our protocol achieves this extremely strong notion of security with only four rounds and relatively short messages, and the parties themselves (including the firewall) only need to perform a small constant number of operations. This compares quite favorably with protocols that are currently implemented in practice (which of course are completely insecure in our setting), and we therefore believe that this protocol can and should be implemented and used in the real world.

This surprising solution, which we describe in detail in Sect. 5.1, uses hashed Diffie-Hellman (similar in spirit to [27]) and bilinear maps. We also use unique signatures to prevent the signatures from becoming a channel themselves.

Rerandomizable Encryption and Active Adversaries. Finally, we return to the question of (necessarily less efficient) protocols based on rerandomizable encryption, but now in the setting of active adversaries. We show how to achieve CCA-security in a single round using rerandomizable RCCA-secure encryption [4]. (See Appendix B.) Indeed, we show that such a primitive is actually *equivalent to* a one-round protocol with a firewall that preserves CCA-security. Such schemes are fairly well-studied, and solutions exist [24,36]. But, our work leads to an interesting open question. Currently known schemes are rerandomizable in the sense that the rerandomization of any valid ciphertext is indistinguishable from a fresh ciphertext, even with access to a decryption oracle. We ask whether these schemes can be made "strongly rerandomizable," in the sense that the same is true even for *invalid* ciphertexts. (See Appendix B for the formal definition.) We show the weaker notion of rerandomizability is equivalent to protocols with firewalls that are secure against functionality-maintaining corruption, while strong rerandomizability is equivalent to security against arbitrary corruption.

1.3 Related Work

Message transmission in the classical setting (i.e., without reverse firewalls) is of course extremely well-studied, and a summary of such work is beyond the scope of this paper. We note, however, that our security definitions for message transmission protocols follow closely Dodis and Fiore [20].

There have been many different approaches to cryptography in the presence of compromise. [31] contains a thorough discussion of many of these (though they naturally do not mention the many relevant papers that appeared simultaneously with or after their publication, such as [2,5,6,15,21,37,38]). In particular, [31] contains a detailed comparison of the reverse-firewall framework with many prior models, showing that RFs generalize much of the prior work on insider attacks and various related notions. Here, we focus on works whose setting or techniques are most similar to our own.

Our work can be viewed as a generalization of that of Bellare, Paterson, and Rogaway [7] in a number of directions. We consider multi-round protocols in which the parties might not share secret keys, and we consider arbitrarily compromised adversaries. In order to get around the very strong restrictions proved in [7], we use the RF framework of [31]. Our techniques are therefore quite different. However, we do use the deterministic encryption scheme of Bellare et al. as part of two of our protocols. (See Appendix A.)

Our work is closely related to Mironov and Stephens-Davidowitz [31], which introduces the reverse-firewalls framework. [31] demonstrate feasibility of this framework by constructing reverse firewalls for parties participating in Oblivious Transfer and Secure Function Evaluation protocols—very strong cryptographic primitives. The fact that such strong primitives can be made secure in this model is quite surprising and bodes well for the reverse-firewalls framework. However, these protocols are very inefficient and therefore mostly of theoretical interest. And, while the primitives considered in [31] have very strong functionality, the security notions that they achieve are rather weak (e.g., security in the semi-honest model). To fulfill the promise of reverse firewalls, we need to consider protocols of more practical importance. We construct much more efficient protocols for widely used primitives with very strong security guarantees. Naturally, we inherit some of the techniques of [31], but we also develop many new ideas.

Bellare and Hoang [5] build on [7] in a different direction, showing how to build deterministic and hedged public-key encryption schemes that are secure against randomness subversion and Algorithm Substitution Attacks. Essentially, they show public-key encryption schemes that are secure even when the sender is compromised, provided that (1) the type of compromise is restricted; and (2) the plaintext itself comes from a high-entropy distribution. These notions of security are much weaker than ours, but they achieve them without the use of an RF. (Recall that [7] implies that semantically secure public-key encryption secure against Algorithm Substitution Attacks is not possible in the classical model.)

Recently, Ateniese, Magri, and Venturi studied reverse firewalls for signature schemes and showed a number of clever solutions [2]. Their work can be considered as complementary to ours, as we are concerned with privacy, while they consider authentication. We also note that our more advanced key-agreement scheme uses unique signatures, and we implicitly rely on the fact that unique signatures have a (trivial) reverse firewall. Indeed, the more general primitive of rerandomizable signatures that Ateniese et al. consider would also suffice for our purposes and might be more efficient in practice.

Our frequent use of rerandomization to "sanitize" messages is very similar to much of the prior work on subliminal channels [9,10,16,17,39], divertible protocols [9,10,32], collusion-free protocols [1,29], etc—particularly the elegant work of Blaze, Bleumer, and Strauss [9] and Alwen, shelat, and Visconti [1]. Again, we refer the reader to [31] for a thorough discussion of these models and their relationship to the reverse-firewall framework.

Finally, we note that some of our study of key agreement is similar to work on key-agreement protocols secure against active insiders, and the study of key control (e.g., [18,26,35]). These works consider key-agreement protocols involving at least three parties, in which one or more of the participants wishes to maliciously fix the key or otherwise subvert the security of the protocol. Some of the technical challenges that we encounter are similar to those encountered in the key control literature, and indeed, the simple commitment-based protocol that we present in Sect. 4.1 can be viewed as a simple instantiation of some of the known (more sophisticated) solutions to the key-control problem (see, e.g., [18]). However, since prior work approached this problem from a different perspective—with three or more parties and without reverse firewalls—our more technical solutions presented in Sect. 5.1 are quite different. In particular, almost all prior work on key agreement focuses on creating protocols that produce a "non-malleable" key, whereas our protocols need some type of malleability specifically to allow the firewall to rerandomize the key. Perhaps surprisingly, we accomplish this without sacrificing security, and our techniques might therefore be of independent interest.

2 Definitions

2.1 Reverse Firewalls

We use the definition of reverse firewalls from [31] (and we refer the reader to [31] for further discussion of the reverse-firewall framework). A reverse firewall $\mathcal{W}$ is a stateful algorithm that maps messages to messages. For a party A and reverse firewall $\mathcal{W}$, we define $\mathcal{W} \circ \mathsf{A}$ as the "composed" party in which $\mathcal{W}$ is applied to the messages that A receives before A "sees them" and the messages that A sends before they "leave the local network of A." $\mathcal{W}$ has access to all public parameters, but not to the private input of A or the output of A. (We can think of $\mathcal{W}$ as an "active router" that sits at the boundary between Alice's private network and the outside world and modifies Alice's incoming and outgoing messages.) We repeat all relevant definitions from [31] below, and we add two new ones.

As in [31], we assume that a cryptographic protocol comes with some functionality or correctness requirements $\mathcal{F}$ and security requirements $\mathcal{S}$. (For example, a functionality requirement $\mathcal{F}$ might require that Alice and Bob output the same thing at the end of the protocol. A security requirement $\mathcal{S}$ might ask that no efficient adversary can distinguish between the transcript of the protocol and a uniformly random string.) Throughout, we use $\bar{\mathsf{A}}$ to represent arbitrary adversarial implementations of party A and $\widetilde{\mathsf{A}}$ to represent functionality-maintaining implementations of A (i.e., implementations of A that still satisfy the functionality requirements of the protocol). For a protocol $\mathcal{P}$ with party A, we write $\mathcal{P}_{\mathsf{A}\to\widetilde{\mathsf{A}}}$ to represent the protocol in which the role of party A is replaced by party $\widetilde{\mathsf{A}}$.

We are only interested in firewalls that themselves maintain functionality. In other words, the *composed party* $\mathcal{W} \circ \mathsf{A}$ should not break the correctness of the protocol. (Equivalently, $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\mathsf{A}}$ should satisfy the same functionality requirements as the underlying protocol $\mathcal{P}$.) We follow [31] in requiring something slightly stronger—reverse firewalls should be "stackable", so that many reverse firewalls composed in series $\mathcal{W} \circ \cdots \circ \mathcal{W} \circ \mathsf{A}$ still do not break correctness.

Definition 1 (Reverse firewall). *A reverse firewall* $\mathcal{W}$ maintains functionality $\mathcal{F}$ for party A in protocol $\mathcal{P}$ *if protocol* $\mathcal{P}$ *satisfies* $\mathcal{F}$*, the protocol* $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\mathsf{A}}$ *satisfies* $\mathcal{F}$*, and the protocol* $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\cdots\circ\mathcal{W}\circ\mathsf{A}}$ *also satisfies* $\mathcal{F}$*. (I.e., we can compose arbitrarily many reverse firewalls without breaking functionality.)*

Of course, a firewall is not interesting unless it provides some benefit. The most natural reason to deploy a reverse firewall is to *preserve* the security of a protocol, even in the presence of compromise. The below definition (which again follows [31]) captures this notion by asking that the protocol obtained by replacing party A with $\mathcal{W} \circ \widetilde{A}$ for an arbitrary corrupted party $\widetilde{A}$ still achieves some notion of security. For example, when we consider message transmission, we will want the firewall to guarantee Alice's privacy against some adversary, even when Alice's own computer has been corrupted.

Definition 2 (Security preservation). *A reverse firewall* strongly preserves security $\mathcal{S}$ for party A in protocol $\mathcal{P}$ *if protocol* $\mathcal{P}$ *satisfies* $\mathcal{S}$*, and for any polynomial-time algorithm* $\bar{\mathsf{A}}$*, the protocol* $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\bar{\mathsf{A}}}$ *satisfies* $\mathcal{S}$*. (I.e., the firewall can guarantee security even when an adversary has tampered with A.)*

A reverse firewall preserves security $\mathcal{S}$ for party A in protocol $\mathcal{P}$ satisfying functionality requirements $\mathcal{F}$ *if protocol* $\mathcal{P}$ *satisfies* $\mathcal{S}$*, and for any polynomial-time algorithm* $\widetilde{\mathsf{A}}$ *such that* $\mathcal{P}_{\mathsf{A}\to\widetilde{\mathsf{A}}}$ *satisfies* $\mathcal{F}$*, the protocol* $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\widetilde{\mathsf{A}}}$ *satisfies* $\mathcal{S}$*. (I.e., the firewall can guarantee security even when an adversary has tampered with A, provided that the tampered implementation does not break the functionality of the protocol.)*

For technical reasons, we will also need a new definition not present in [31]. We wish to show generic composition theorems, allowing us to construct a message-transmission protocol with secure reverse firewall from any key-agreement protocol with its own firewalls. In order to accomplish this, we will need the notion of *detectable failure.* Essentially, a protocol fails detectably if we can distinguish between transcripts of valid runs of the protocol and invalid transcripts. For simplicity, we assume that honest parties always output $\perp$ when they receive a malformed message (e.g., when a message that should be a pair of group elements is not a pair of group elements). While the general notion of validity is a bit technical, we will use it in very straightforward ways. (E.g., transcripts will be valid if and only if a commitment is properly opened and a certain signature is valid.)

Definition 3 (Valid transcripts). *A sequence of bits r and private input I* generate transcript $\mathcal{T}$ in protocol $\mathcal{P}$ *if a run of the protocol $\mathcal{P}$ with input I in which the parties' coin flips are taken from r results in the transcript $\mathcal{T}$. A transcript $\mathcal{T}$ is a* valid transcript for protocol $\mathcal{P}$ *if there is a sequence r and private input I generating $\mathcal{T}$ such that no party outputs $\perp$ at the end of the run. (Here, we assume that the public input is part of the transcript.) A protocol* has unambiguous transcripts *if for any valid transcript $\mathcal{T}$, there is no possible input I and coins r generating $\mathcal{T}$ that results in a party outputting $\perp$. (In other words, a valid transcript never results from a failed run of the protocol.)*

Definition 4 (Detectable failure). *A reverse firewall $\mathcal{W}$* detects failure for party A in protocol $\mathcal{P}$ *if*

- $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\mathsf{A}}$ *has unambiguous transcripts;*
- *the firewall $\mathcal{W}$ outputs the special symbol $\perp$ when run on any transcript that is not valid for $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\mathsf{A}}$; and*
- *there is a polynomial-time deterministic algorithm that decides whether a transcript $\mathcal{T}$ is valid for $\mathcal{P}_{\mathsf{A}\to\mathcal{W}\circ\mathsf{A}}$.*

We will also need the notion of *exfiltration resistance*, introduced in [31]. Intuitively, a reverse firewall is exfiltration resistant if "no corrupt implementation of Alice can leak information through the firewall." We say that it is exfiltration resistant for Alice against Bob if Alice cannot leak information to Bob through the firewall, and we say that it is exfiltration resistant against eavesdroppers (or just exfiltration resistant) if Alice cannot leak information through the firewall to an adversary that is only given access to the protocol transcript.

The second definition below (which uses the notion of valid transcripts) is new to this paper and is necessary for our composition theorems.

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

Fig. 1. $\mathsf{LEAK}(\mathcal{P}, \mathsf{A}, \mathsf{B}, \mathcal{W}, \lambda)$, the exfiltration resistance security game for a reverse firewall $\mathcal{W}$ for party A in protocol $\mathcal{P}$ against party B with input I. $\mathcal{E}$ is the adversary, λ the security parameter, $S_{\bar{B}}$ the state of $\bar{B}$ after the run of the protocol, I valid input for $\mathcal{P}$, and $\mathcal{T}^*$ is the transcript resulting from a run of the protocol $mathcalP_{\mathsf{A}\to\mathsf{A}^*,\mathsf{B}\to\bar{B}}$ with input I.

Definition 5 (Exfiltration resistance). *A reverse firewall is* exfiltration resistant for party A against party B in protocol $\mathcal{P}$ satisfying functionality $\mathcal{F}$ *if no PPT algorithm* $\mathcal{E}$ *with output circuits* $\widetilde{\mathsf{A}}$ *and* $\widetilde{\mathsf{B}}$ *such that* $\mathcal{P}_{\mathsf{A}\to\widetilde{\mathsf{A}}}$ *and* $\mathcal{P}_{\mathsf{B}\to\widetilde{\mathsf{B}}}$ *satisfy* $\mathcal{F}$ *has non-negligible advantage in* $\mathsf{LEAK}(\mathcal{P}, \mathsf{A}, \mathsf{B}, \mathcal{W}, \lambda)$. *If* B *is empty, then we simply say that the firewall is* exfiltration resistant.

A reverse firewall is exfiltration resistant for party A against party B in protocol $\mathcal{P}$ with valid transcripts *if no PPT algorithm* $\mathcal{E}$ *with output circuits* $\widetilde{\mathsf{A}}$ *and* $\widetilde{\mathsf{B}}$ *such that* $\mathcal{P}_{\mathsf{A}\to\widetilde{\mathsf{A}}}$ *and* $\mathcal{P}_{\mathsf{B}\to\widetilde{\mathsf{B}}}$ *produce valid transcripts for* $\mathcal{P}$ *has non-negligible advantage in* $\mathsf{LEAK}(\mathcal{P}, \mathsf{A}, \mathsf{B}, \mathcal{W}, \lambda)$. *If* B *is empty, then we simply say that the firewall is* exfiltration resistant with valid transcripts.

A reverse firewall is strongly exfiltration resistant for party A against party B in protocol $\mathcal{P}$ *if no PPT adversary* $\mathcal{E}$ *has non-negligible advantage in* $\mathsf{LEAK}(\mathcal{P}, \mathsf{A}, \mathsf{B}, \mathcal{W}, \lambda)$. *If* B *is empty, then we say that the firewall is* strongly exfiltration resistant.

2.2 Message-Transmission Protocols

A *message-transmission protocol* is a two-party protocol in which one party, Alice, is able to communicate a plaintext message to the other party, Bob. (For simplicity, we only formally model the case in which Alice wishes to send a single plaintext to Bob per run of the protocol, but this of course naturally extends to a more general case in which Alice and Bob wish to exchange many plaintext messages.) We consider two notions of security for such messages. First, we consider *CPA security*, in which the adversary must distinguish between the transcript of a run of the protocol in which Alice communicates the plaintext m_0 to Bob and the transcript with which Alice communicates m_1 to Bob, where m_0 and m_1 are adversarially chosen plaintexts. (Even in this setting, we allow the adversary to start many concurrent runs of the protocol with adaptively chosen plaintexts.) Our strongest notion of security is *CCA security* in which the adversary may "feed" the parties any messages and has access to a decryption oracle. Our security definitions are similar in spirit to [20], but adapted for our setting.

Session Ids. Throughout this paper, we consider protocols that may be run concurrently many times between the same two parties. In order to distinguish one run of a protocol from another, we therefore "label" each run with a unique session id, denoted sid. We view sid as an implicit part of every message, and we often ignore sid when it is not important. Our parties and firewalls are stateful, and we assume that the parties and the firewall maintain a list of the relevant session ids, together with any information that is relevant to continue the run of the protocol corresponding to sid (such as the number of messages sent so far, any values that need to be used later in the protocol, etc.). We typically suppress explicit reference to these states. In our security games, the adversary may choose the value sid for each run of the protocol, provided that each party has a unique run for each session sid. (In fact, it does not even make sense for the adversary to use the same sid for two different runs of the protocol with the same party, as this party will necessarily view any calls with the same sid as corresponding to a single run of the protocol. However, as is clear from our security games, an active adversary may maintain two separate runs of a protocol with two different parties but the same sid.) In practice, sid can be a simple counter or any other nonce (perhaps together with any practical information necessary for communication, such as IP addresses). We note in passing that, in the setting of reverse firewalls, a counter is preferable to, e.g., a random nonce to avoid providing a channel through sid, but such concerns are outside of our model and the scope of this paper.

The definition below makes the above formal and provides us with some useful terminology.

Definition 6 (Message-transmission protocol). *A* message-transmission protocol *is a two-party protocol in which one party, Alice, receives as input a plaintext m from some plaintext space $\mathcal{M}$. The protocol is* correct *if for any input $m \in \mathcal{M}$, Bob's output is always m.*

We represent the protocol by four algorithms $\mathcal{P} = (\mathsf{setup}, \mathsf{next}_\mathsf{A}, \mathsf{next}_\mathsf{B}, \mathsf{return}_\mathsf{B})$. setup takes as input 1^λ, where λ is the security parameter, and returns the starting states for each party, $S_\mathsf{A}, S_\mathsf{B}$, which consist of both private input, σ_A and σ_B respectively, and public input π. Each party's next procedure is a stateful algorithm that takes as input sid and an incoming message, updates the party's state, and returns an outgoing message. The $\mathsf{return}_\mathsf{B}$ procedure takes as input Bob's state S_B and sid and returns Bob's final output.

We say that a message-transmission protocol is

– unkeyed *if setup does not return any private input σ_A or σ_B;*
– singly keyed *if setup returns private input σ_B for Bob but none for Alice;*
– publicly keyed *if setup returns private input for both parties σ_A and σ_B, but these private inputs are independently distributed; and*
– privately keyed *if setup returns private input for both parties whose distributions are dependent.*

When we present protocols, we will drop the formality of defining explicit functions $\mathcal{P} = (\mathsf{setup}, \mathsf{next}_\mathsf{A}, \mathsf{next}_\mathsf{B}, \mathsf{return}_\mathsf{B})$ and states for the parties, preferring instead to use diagrams as in Fig. 4. But, this formulation is convenient for our security definitions. In particular, we present the relevant subprocedures for our security games in Fig. 2. An adversary plays the game depicted in Fig. 2 by first calling initialize (receiving as output π) and then making various calls to the other subprocedures. Each time it calls a subprocedure, it receives any output from the procedure. The game ends when the adversary calls $\mathsf{finalize}$, and the adversary wins if and only if the output of $\mathsf{finalize}$ is one.

The below definitions capture formally the intuitive notions of security that we presented above. In particular, the CPA security definition allows the adversary to start arbitrarily many concurrent runs of the protocol with adversarial input, but it does not allow the adversary to change the messages sent by the

```
proc. initialize($1^\lambda$)
$(\sigma_A, \sigma_B, \pi) \xleftarrow{\$} \mathsf{setup}(1^\lambda)$
$S_A \leftarrow (\sigma_A, \pi)$; $S_B \leftarrow (\sigma_B, \pi)$
$\mathsf{sid}^* \leftarrow \perp$; $\mathsf{compromised} \leftarrow \mathsf{false}$
$b \xleftarrow{\$} \{0, 1\}$
OUTPUT $\pi$

proc. finalize($b^*$)
IF $b = b^*$, RETURN 1
ELSE, RETURN 0

proc. start-run($\mathsf{sid}, m$)
IF $\mathsf{sid} \notin S_A$, $S_A.\mathsf{add}(\mathsf{sid}, m)$

proc. start-challenge($\mathsf{sid}, m_0, m_1$)
IF $\mathsf{sid} \notin S_A$ AND $\mathsf{sid}^* = \perp$,
   $\mathsf{sid}^* \leftarrow \mathsf{sid}$
   $S_A.\mathsf{add}(\mathsf{sid}, m_b)$

proc. get-next_A($\mathsf{sid}, M$)
IF compromised,
    OUTPUT $\perp$
OUTPUT $\mathsf{next}_A(S_A, \mathsf{sid}, M)$

proc. get-next_B($\mathsf{sid}, M$)
IF compromised,
    OUTPUT $\perp$
OUTPUT $\mathsf{next}_B(S_B, \mathsf{sid}, M)$

proc. get-output_B($\mathsf{sid}$)
IF $\mathsf{sid} = \mathsf{sid}^*$ OR compromised,
    OUTPUT $\perp$
OUTPUT $\mathsf{return}_B(S_B, \mathsf{sid})$

proc. get-secrets
compromised $\leftarrow$ true
OUTPUT $(\sigma_A, \sigma_B)$
```

Fig. 2. Procedures used to define security for message-transmission protocol $\mathcal{P} = (\mathsf{setup}, \mathsf{next}_A, \mathsf{next}_B)$. An adversary plays this game by first calling initialize and then making various oracle calls. The game ends when the adversary calls finalize, and the output of finalize is one if the adversary wins and zero otherwise.

two parties or to send its own messages. We also define forward secrecy, which requires that security hold even if the parties' secret keys may be leaked to the adversary.

Definition 7 (Message-transmission security). *A message-transmission protocol is called*

- chosen-plaintext secure *(CPA-secure) if no PPT adversary has non-negligible advantage in the game presented in Fig. 2 when* $\mathsf{get\text{-}next}_A(\mathsf{sid}, M)$ *and* $\mathsf{get\text{-}next}_B(\mathsf{sid}, M)$ *output* $\perp$ *unless this is the first* get-next *call with this* sid *or* M *is the output from the previous* $\mathsf{get\text{-}next}_A$ *call with the same* sid *or the previous* $\mathsf{get\text{-}next}_B$ *with the same* sid *respectively (i.e., the adversary is passive); and*
- chosen-ciphertext secure *(CCA-secure) if no PPT adversary has non-negligible advantage in the game presented in Fig. 2 with access to all oracles.*

We say that the protocol is chosen-plaintext (resp. chosen-ciphertext) secure without forward secrecy *if the above holds without access to the* get-secrets *oracle.*

We note that it does not make sense to consider chosen-ciphertext security when Bob may be corrupted. In this case, the output of $\mathsf{get\text{-}output}_B$ could be arbitrary. (Note that the firewall can potentially "sanitize" Bob's *messages*, but it of course does not have access to his *output*.) We therefore only consider firewalls that preserve CPA security for Bob.

2.3 Key Agreement

Key-agreement protocols will play a central role in our constructions, so we now provide a definition of key agreement that suffices for our purposes. Our notion of key agreement closely mirrors the definitions from the previous section.

Definition 8 (Key agreement). *A* key-agreement protocol *is represented by five algorithms,* $\mathcal{P} = (\mathsf{setup}, \mathsf{next}_\mathsf{A}, \mathsf{next}_\mathsf{B}, \mathsf{return}_\mathsf{A}, \mathsf{return}_\mathsf{B})$. setup *takes as input* 1^λ*, where* λ *is the security parameter and returns the starting states for each party,* $S_\mathsf{A}, S_\mathsf{B}$*, which consists of public input* π *and the private input for each party* σ_A *and* σ_B*. Each party's* next *procedure is a stateful algorithm that takes as input* sid *and an incoming message, updates the party's state, and returns an outgoing message. Each party's* return *procedure takes as input the relevant party's state and* sid *and returns the party's final output from some key space* $\mathcal{K}$ *or* $\perp$*. We also allow* auxiliary input aux *to be added to Alice's state before the first message of a protocol is sent.*

The protocol is correct *if Alice and Bob always output the same thing at the end of the run of a protocol for any random coins and auxiliary input* aux*.*

We say that a key-agreement protocol is

– unkeyed *if* setup *does not return any private input* σ_A *or* σ_B*;*
– singly keyed *if* setup *returns private input* σ_B *for Bob but no private input* σ_A *for Alice; and*
– publicly keyed *if* setup *returns private input for both parties* σ_A *and* σ_B*.*

Definition 9 (Key-agreement security). *A key-agreement protocol is*

– secure against passive adversaries *if no probabilistic polynomial-time adversary has non-negligible advantage in the game presented in Fig. 3 when* $\mathsf{get\text{-}next}_\mathsf{A}(\mathsf{sid}, M)$ *and* $\mathsf{get\text{-}next}_\mathsf{B}(\mathsf{sid}, M)$ *output* $\perp$ *unless this is the first* $\mathsf{get\text{-}next}$ *call with this* sid *or* M *is the output from the previous* $\mathsf{get\text{-}next}_\mathsf{B}$ *call with the same* sid *or the previous* $\mathsf{get\text{-}next}_\mathsf{A}$ *call with the same* sid *respectively (i.e., the adversary is passive);*
– secure against active adversaries for Alice *if no probabilistic polynomial-time algorithm has non-negligible advantage in the game presented in Fig. 3 without access to the* $\mathsf{get\text{-}output}_\mathsf{A}$ *oracle;*
– secure against active adversaries for Bob *if no probabilistic polynomial-time algorithm has non-negligible advantage in the game presented in Fig. 3 without access to the* $\mathsf{get\text{-}output}_\mathsf{B}$ *oracle; and*
– secure against active adversaries *if it is secure against active adversaries for both Bob and Alice; and*

```
proc. initialize($1^\lambda$)
$(\sigma_A, \sigma_B, \pi) \xleftarrow{\$} \mathsf{setup}(1^\lambda)$
$S_A \leftarrow (\sigma_A, \pi)$
$S_B \leftarrow (\sigma_B, \pi)$
$\mathsf{sid}^* \leftarrow \bot$
compromised $\leftarrow$ false
$b \xleftarrow{\$} \{0,1\}$
OUTPUT $\pi$

proc. finalize($b^*$)
IF $b = b^*$,
    RETURN 1
ELSE, RETURN 0

proc. start-run(sid, aux)
IF $\mathsf{sid} \notin S_A$,
   $S_A$.add(sid, aux)

proc. start-challenge(sid, aux)
IF $\mathsf{sid}^* = \bot$ AND $\mathsf{sid} \notin S_A$,
    $\mathsf{sid}^* \leftarrow \mathsf{sid}$
    $\mathcal{R}_{\mathsf{sid}^*} \xleftarrow{\$} \mathcal{K}$
    $S_A$.add(sid, aux)

proc. get-next$_A$(sid, $M$)
IF NOT compromised,
    OUTPUT $\mathsf{next}_A(S_A, \mathsf{sid}, M)$

proc. get-next$_B$(sid, $M$)
IF NOT compromised,
    OUTPUT $\mathsf{next}_B(S_B, \mathsf{sid}, M)$

proc. get-output$_A$(sid)
IF compromised, OUTPUT $\bot$
IF $\mathsf{sid} = \mathsf{sid}^*$ AND $b = 0$,
    IF $\mathsf{return}_A(S_A, \mathsf{sid}) = \bot$, OUTPUT $\bot$
    ELSE, OUTPUT $R_{\mathsf{sid}}$
ELSE, OUTPUT $\mathsf{return}_A(S_A, \mathsf{sid})$

proc. get-output$_B$(sid)
IF compromised, OUTPUT $\bot$
IF $\mathsf{sid} = \mathsf{sid}^*$ AND $b = 0$,
    IF $\mathsf{return}_B(S_B, \mathsf{sid}) = \bot$, OUTPUT $\bot$
    ELSE, OUTPUT $R_{\mathsf{sid}}$
ELSE, OUTPUT $\mathsf{return}_B(S_B, \mathsf{sid})$

proc. get-secrets
compromised $\leftarrow$ true
OUTPUT $(\sigma_A, \sigma_B)$
```

Fig. 3. Procedures used to define security for key-agreement protocol $\mathcal{P} = (\mathsf{setup}, \mathsf{next}_A, \mathsf{next}_B, \mathsf{return}_A, \mathsf{return}_B)$. An adversary plays this game by first calling initialize and then making various oracle calls. The game ends when the adversary calls finalize, and the output of finalize is one if the adversary wins and zero otherwise. We suppress the auxiliary input aux when it is irrelevant.

– authenticated for Bob *if no probabilistic polynomial-time algorithm playing the game presented in Fig. 3 can output a valid transcript with corresponding session id* sid *unless* $\mathsf{return}_B(S_B, \mathsf{sid}) \neq \bot$ *or* compromised = true. *(I.e., it is hard to find a valid transcript unless Bob returns a key.) Furthermore, if the transcript is valid and* $\mathsf{get\text{-}output}_A(\mathsf{sid}) \neq \bot$ *then* $\mathsf{get\text{-}output}_A(\mathsf{sid}) = \mathsf{return}_B(\mathsf{sid})$. *(I.e., if the transcript is valid and Alice outputs a key, then Bob outputs the same key.)*

Note that these definitions are far from standard. In particular, in the case of active adversaries, we define security for Alice in terms of the keys that *Bob* outputs and security for Bob in terms of the keys that *Alice* outputs. This may seem quite counterintuitive. But, in our setting, we are worried that Alice may be corrupted. In this case, we cannot hope to restrict Alice's output after she receives invalid messages. (The firewall can modify Alice's *messages*, but not her *output.*) So, the best we can hope for is that the firewall prevents a tampered implementation of Alice (together with an active adversary) from "tricking" Bob into returning an insecure key.

3 A Two-Round Protocol from Rerandomizable Encryption

We first consider the simple case of CPA-secure two-round schemes in which the first message is a public key chosen randomly by Bob and the second message is an encryption of Alice's plaintext under this public key. Figure 4 shows the protocol.

In order to provide a reverse firewall for Alice in this protocol, the encryption scheme must be *rerandomizable*. In order to provide a reverse firewall for Bob, the scheme must be *key malleable*. Intuitively, a scheme is key malleable if a third party can "rerandomize" a public key and map ciphertexts under the "rerandomized" public key to ciphertexts under the original public key. We include formal definitions in the full version, and we observe there that ElGamal encryption suffices [22].

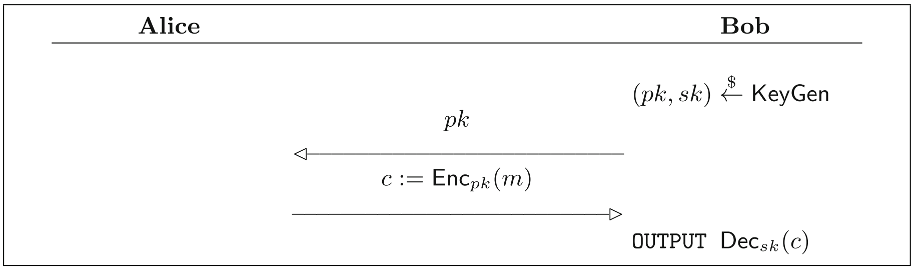
Alice
Bob
$(pk, sk) \overset{\$}{\leftarrow} \mathsf{KeyGen}$
pk
$c := \mathsf{Enc}_{pk}(m)$
OUTPUT $\mathsf{Dec}_{sk}(c)$

Fig. 4. Two round message-transmission protocol using the public-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

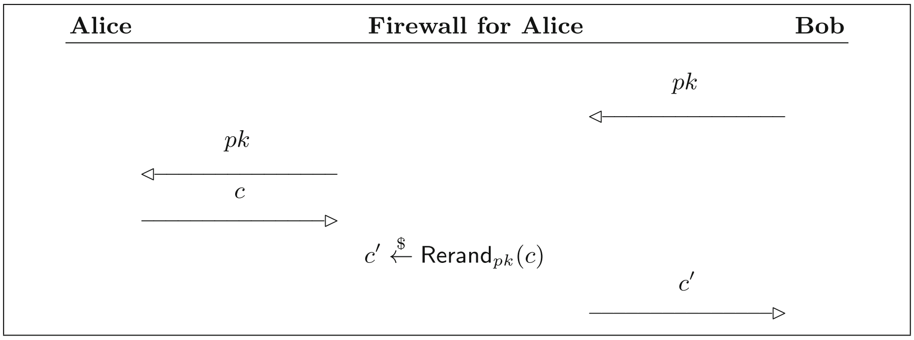

Fig. 5. Reverse firewall for Alice for the protocol shown in Fig. 5 that works if the encryption scheme is rerandomizable.

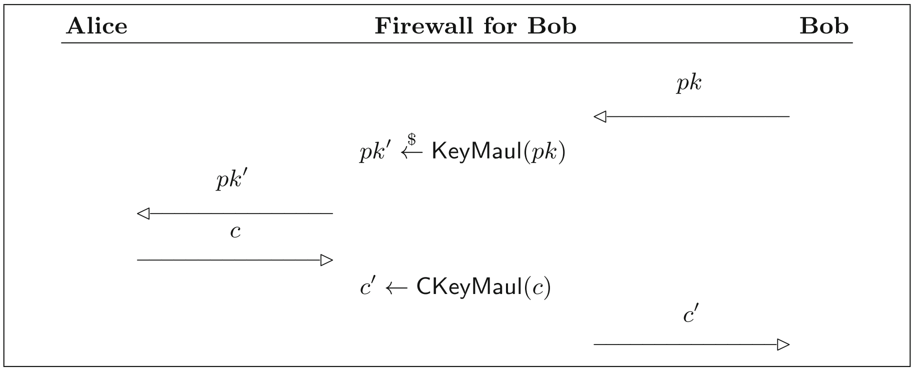
Alice
Firewall for Bob
Bob
pk
$pk' \overset{\$}{\leftarrow} \mathsf{KeyMaul}(pk)$
pk'
c
$c' \leftarrow \mathsf{CKeyMaul}(c)$
c'

Fig. 6. Reverse firewall for Bob for the protocol shown in Fig. 4 that works if the encryption scheme is key malleable. We suppress the randomness r used as input to $\mathsf{KeyMaul}$ and $\mathsf{CKeyMaul}$.

If the underlying encryption scheme in Fig. 4 is rerandomizable, then we can build a reverse firewall for Alice as in Fig. 5. If it is key malleable, then we can build a reverse firewall for Bob as in Fig. 6. The following theorem shows that this protocol and its reverse firewalls are secure. The simple proof is included in the full version [22].

Theorem 1. *The unkeyed message-transmission protocol shown in Fig. 4 is CPA-secure if the underlying encryption scheme is semantically secure. If the scheme is also (strongly) rerandomizable, then the reverse firewall shown in Fig. 5 (strongly) preserves security for Alice and (strongly) resists exfiltration for Alice. If the scheme is key malleable, then the reverse firewall shown in Fig. 6 maintains functionality for Bob, strongly preserves Bob's security, and strongly resists exfiltration for Bob against Alice.*

3.1 Hybrid encryption fails.

A major drawback of the above scheme is that it requires public-key operations of potentially very long plaintexts, which can be very inefficient in practice. A common solution in the classical setting is to use *hybrid encryption*, in which $\mathsf{Enc}_{pk}(m)$ is replaced by $(\mathsf{Enc}_{pk}(rk), \mathsf{SEnc}_{rk}(m))$, where SEnc is some suitable symmetric-key encryption scheme and rk is a freshly chosen uniformly random key for SEnc. However, if we simply replace the public-key encryption in Fig. 4 with the corresponding hybrid-key encryption scheme, then this fails spectacularly. For example, a tampered version of Alice $\widetilde{\mathsf{A}}$ can choose some *fixed* secret key rk^* and send the message $(\mathsf{Enc}_{pk}(rk^*), \mathsf{SEnc}_{rk^*}(m))$. If rk^* is a valid key, then $\widetilde{\mathsf{A}}$ maintains functionality, but an adversary that knows rk^* can of course read any messages that Alice sends.

So, in order for such a protocol to have a secure reverse firewall, the RF must be able to maul the encrypted key $\mathsf{Enc}_{pk}(rk)$ into $\mathsf{Enc}_{pk}(rk')$ for some

rk' and then convert the encrypted plaintext $\mathsf{SEnc}_{rk}(m)$) into an encryption under this new key $\mathsf{SEnc}_{rk'}(m)$). In particular, the symmetric-key scheme must be "key malleable." Unfortunately, such a scheme implies public-key encryption. Therefore, our supposed "symmetric-key" scheme actually must use public-key primitives. So, hybrid encryption buys us nothing. (In the full version [22], we give a proof sketch.)

Remark (Informal). *Any key-malleable symmetric-key encryption scheme implies public-key encryption.*

4 A Solution Using Key Agreement

In this section, we remain in the setting in which neither Alice nor Bob has a public key, so we are still interested in CPA security. (We address CCA security in the next section.) The protocol from Sect. 3 works, but it requires a public-key operation on the plaintext, which may be very long. In practice, this can be very inefficient. And, we showed in Sect. 3.1 that one common solution to this problem in the classical setting, hybrid encryption, seems hopeless with reverse firewalls because it allows Alice to *choose* a key rk that will be used to encrypt the plaintext—thus allowing a tampered version of Alice to "choose a bad key."

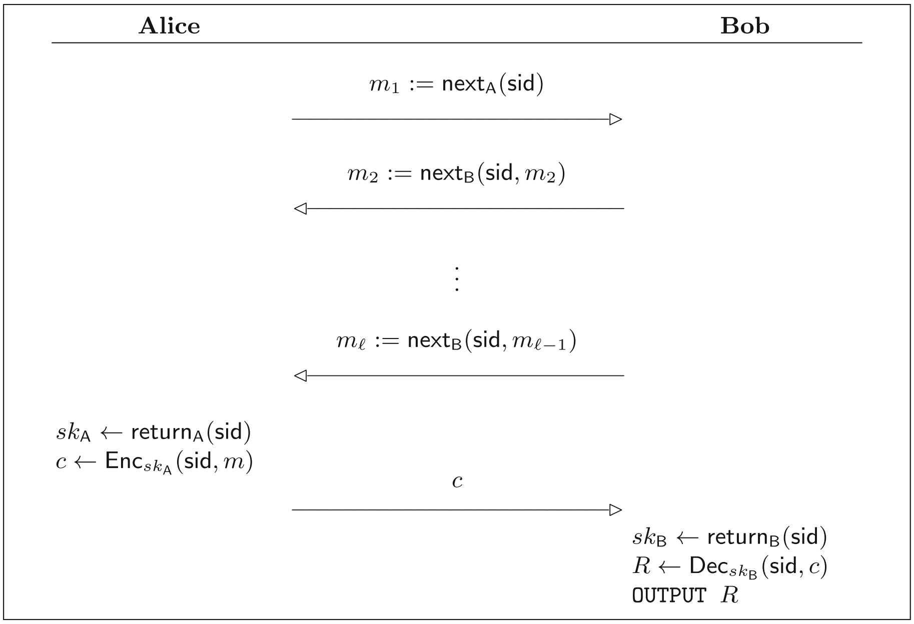
Alice
Bob
$m_1 := \mathsf{next}_\mathsf{A}(\mathsf{sid})$
$m_2 := \mathsf{next}_\mathsf{B}(\mathsf{sid}, m_2)$
$\vdots$
$m_\ell := \mathsf{next}_\mathsf{B}(\mathsf{sid}, m_{\ell-1})$
$sk_\mathsf{A} \leftarrow \mathsf{return}_\mathsf{A}(\mathsf{sid})$
$c \leftarrow \mathsf{Enc}_{sk_\mathsf{A}}(\mathsf{sid}, m)$
c
$sk_\mathsf{B} \leftarrow \mathsf{return}_\mathsf{B}(\mathsf{sid})$
$R \leftarrow \mathsf{Dec}_{sk_\mathsf{B}}(\mathsf{sid}, c)$
OUTPUT R

Fig. 7. The message-transfer protocol obtained by combining a key-agreement scheme (setup, next_A, next_B, $\mathsf{return}_\mathsf{A}$, $\mathsf{return}_\mathsf{B}$) and a nonce-based encryption scheme, (Enc, Dec).

So, we instead consider an alternative common solution to this efficiency problem: key agreement followed by symmetric-key encryption. (See Fig. 7.) As in Appendix A, we use a nonce-based encryption scheme with unique ciphertexts. We can view this as a modification of hybrid encryption in which "Alice and Bob together choose the key rk" that will be used to encrypt the plaintext. More importantly from our perspective, the messages that determine the key will go through the firewall. As an added benefit, once a key is established, Alice can use it to efficiently send multiple messages, not just one, without any additional public-key operations (though we do not model this here).

The composition theorem below shows that this protocol can in fact have a reverse firewall for both parties, provided that the key-agreement protocol itself has a reverse firewall that satisfies some suitable security requirements. See the full version for the proof. In the next section, we construct such a protocol.

Theorem 2 (Composition theorem for CPA security). *Let $\mathcal{W}_{\mathsf{A}}$ and $\mathcal{W}_{\mathsf{B}}$ be reverse firewalls in the underlying key-agreement protocol in Fig. 7 for Alice and Bob respectively. Let $\mathcal{W}_{\mathsf{A}}^*$ be the firewall for Alice in the full protocol in Fig. 7 obtained by applying $\mathcal{W}_{\mathsf{A}}$ to the key-agreement messages and then letting the last message through if $\mathcal{W}_{\mathsf{A}}$ does not output $\perp$ and replacing the last message by $\perp$ otherwise. Let $\mathcal{W}_{\mathsf{B}}^*$ be the firewall for Bob in the full protocol in Fig. 7 obtained by applying $\mathcal{W}_{\mathsf{B}}$ to the key-agreement messages and simply letting the last message through. Then,*

1. *the protocol in Fig. 7 is CPA-secure if the underlying key-agreement protocol is secure against passive adversaries and the underlying nonce-based encryption scheme is CPA-secure;*
2. *$\mathcal{W}_{\mathsf{B}}^*$ preserves CPA security if $\mathcal{W}_{\mathsf{B}}$ preserves security of the key-agreement protocol; and*
3. *$\mathcal{W}_{\mathsf{A}}^*$ preserves CPA security if the encryption scheme has unique ciphertexts and $\mathcal{W}_{\mathsf{A}}$ preserves semantic security and is exfiltration resistant against Bob.*

Finally, we note that strong security preservation is not possible for this protocol (at least for Alice). (We include a proof sketch in the full version [22].)

Remark 1 (Informal). There is no reverse firewall for Alice in the protocol illustrated in Fig. 7 that maintains functionality and strongly preserves Alice's security.

4.1 Key Agreement Secure Against Passive Adversaries

Theorem 2 motivates the study of unkeyed key-agreement protocols with reverse firewalls that preserve security against passive adversaries. In the classical setting (i.e., without reverse firewalls), the canonical example is the celebrated key-agreement protocol of Diffie and Hellman [19], shown in Fig. 8, whose security follows immediately from the hardness of DDH over the base group G. We use this as an example to illustrate the basic idea of a reverse firewall in the key-agreement setting.

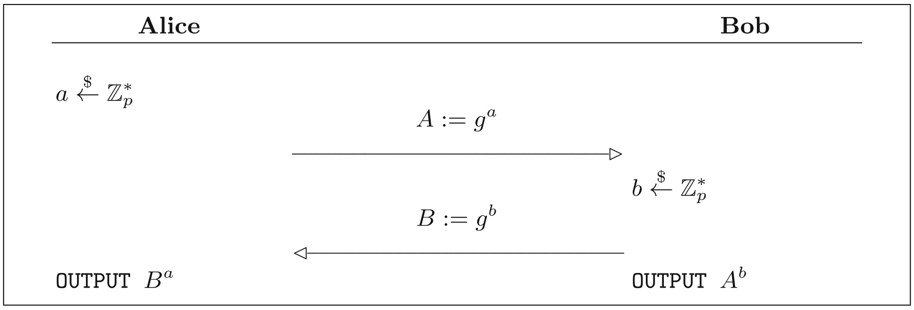
Alice
Bob
$a \xleftarrow{\$} \mathbb{Z}_p^*$
$A := g^a$
$b \xleftarrow{\$} \mathbb{Z}_p^*$
$B := g^b$
OUTPUT B^a
OUTPUT A^b

Fig. 8. Diffie-Hellman key agreement over a group G of prime order p with generator g.

Diffie-Hellman key agreement has a simple reverse firewall for Alice, which raises both messages to a single random power, $\alpha \in \mathbb{Z}_p^*$. We present this reverse firewall in Fig. 9. Note that this firewall effectively replaces Alice's message with a uniformly random message. Security then follows from the security of the underlying protocol, since the transcript and resulting key in the two cases are distributed identically.

But, this protocol cannot have a reverse firewall that maintains correctness and preserves security for Bob, as we described in the introduction. In particular, we run the risk that one party has the ability to selectively reject keys.

To solve this problem, we add an additional message to the beginning of the protocol in which Bob commits to the message that he will send later. Of course, in order to permit a secure firewall, the commitment scheme itself must be both malleable (so that the firewall can rerandomize the underlying message that Bob has committed to, mapping a commitment of B to a commitment of B^α) and rerandomizable (so that the randomness used by Bob to commit and open will not leak any information about his message). To achieve our strongest level of security,

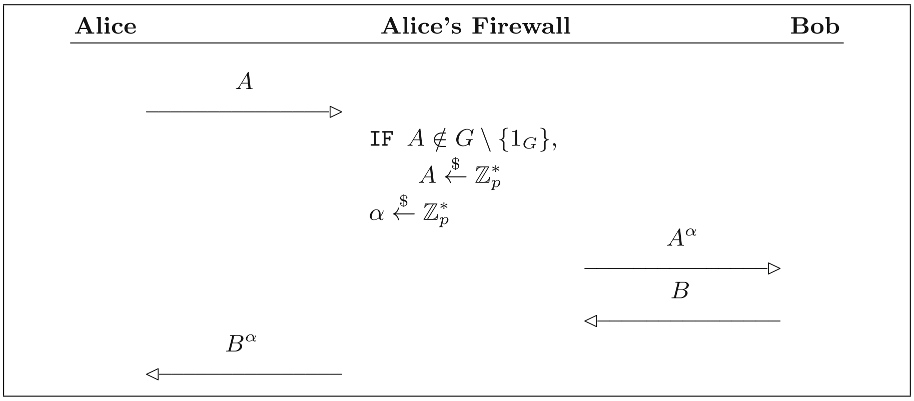
Alice
Alice's Firewall
Bob
A
IF $A \notin G \setminus \{1_G\}$,
$A \xleftarrow{\$} \mathbb{Z}_p^*$
$\alpha \xleftarrow{\$} \mathbb{Z}_p^*$
A^α
B
B^α

Fig. 9. Reverse firewall for Alice in the protocol from Fig. 8.

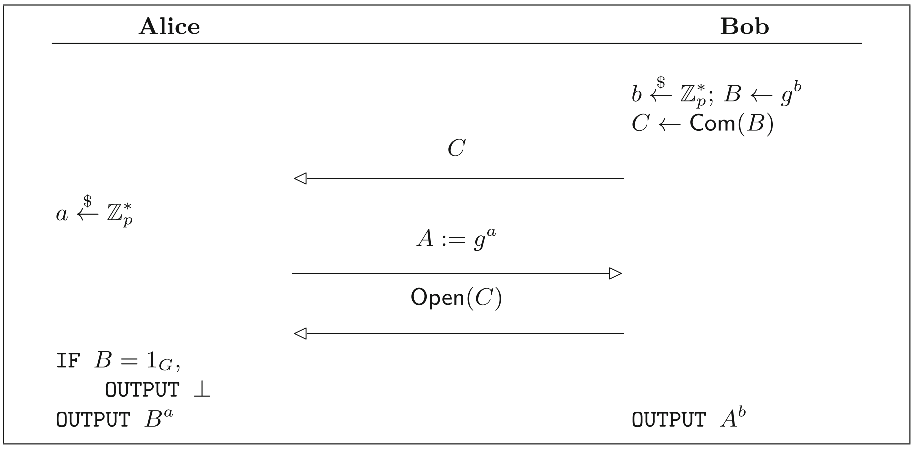
Alice
Bob
$b \xleftarrow{\$} \mathbb{Z}_p^*$; $B \leftarrow g^b$
$C \leftarrow \mathsf{Com}(B)$
C
$a \xleftarrow{\$} \mathbb{Z}_p^*$
$A := g^a$
$\mathsf{Open}(C)$
IF $B = 1_G$,
OUTPUT $\perp$
OUTPUT B^a
OUTPUT A^b

Fig. 10. A variant of Diffie-Hellman key agreement over a group G of prime order p with public generator g. $(\mathsf{Com}, \mathsf{Open})$ is a commitment scheme.

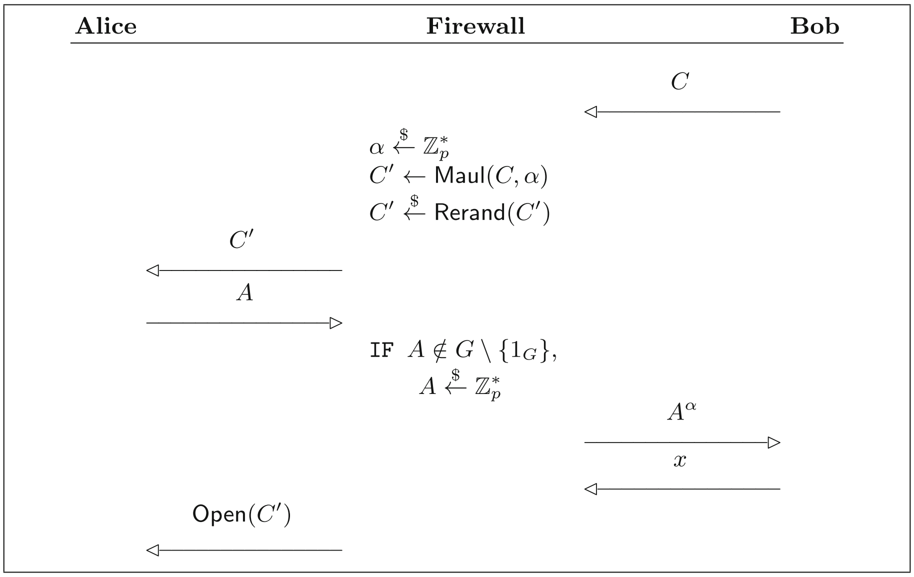
Alice
Firewall
Bob
C
$\alpha \xleftarrow{\$} \mathbb{Z}_p^*$
$C' \leftarrow \mathsf{Maul}(C, \alpha)$
$C' \xleftarrow{\$} \mathsf{Rerand}(C')$
C'
A
IF $A \notin G \setminus \{1_G\}$,
$A \xleftarrow{\$} \mathbb{Z}_p^*$
A^α
x
$\mathsf{Open}(C')$

Fig. 11. Reverse firewall for either Alice or Bob in the protocol from Fig. 8. $\mathsf{Maul}(C, \alpha)$ takes a commitment $C = \mathsf{Com}(B)$ and converts it into a commitment of B^α. $\mathsf{Rerand}(C)$ takes a commitment $C = \mathsf{Com}(B)$ and converts it into a uniformly random commitment of B. We assume that a rerandomized and mauled commitment can be opened with access to an opening of the original commitment (and the randomness used in the rerandomization and mauling).

we also need the scheme to be statistically hiding and for each commitment to have a unique opening for a given message. (These requirements are easily met in practice. For example, a simple variant of the Pedersen commitment suffices [33]. For completeness, we present such a scheme in the full version [22].) The protocol is shown in Fig. 10. In Fig. 11, we present a single reverse firewall for this protocol that happens to work for either party. (Each party would need to deploy its own version of this firewall to guarantee its own security. It just happens that each party's firewall would have the same "code.")

In the full version, we prove the following theorem [22].

Theorem 3. *The protocol in Fig. 10 is secure against passive adversaries if DDH is hard in G. The reverse firewall $\mathcal{W}$ in Fig. 11 is functionality maintaining. If the commitment scheme is statistically hiding, then $\mathcal{W}$ preserves security for Alice and is strongly exfiltration resistant against Bob. If the commitment scheme is computationally binding, then $\mathcal{W}$ is exfiltration resistant for Bob against Alice and preserves security for Bob. $\mathcal{W}$ also detects failure for both parties.*

5 CCA-security Using Key Agreement

In the setting of the previous section, with no public-key infrastructure, it is trivially impossible to achieve CCA-security. (An adversary can simply "pretend to be Bob" and read Alice's plaintext.) In this section, we show that a CCA-secure message-transmission protocol with reverse firewalls does in fact exist in the publicly keyed setting. In particular, below, we give the CCA analogue of Theorem 2, showing that a key-agreement protocol that is secure against active adversaries and has sufficiently secure reverse firewalls together with a symmetric-key encryption scheme suffices. As in the previous section, this key-agreement-based protocol has the additional benefit that it is efficient, in the sense that it does not apply public-key operations to the plaintext. In Sect. 5.1, we construct a key-agreement protocol that suffices.

We now present our stronger composition theorem. (Recall that Bob's reverse firewall can only preserve CPA security. Such a firewall is already given by Theorem 2, so we do not repeat this here.) See the full version for the proof [22].

Theorem 4 (Composition theorem for CCA security). *Define $\mathcal{W}_{\mathsf{A}}$ and $\mathcal{W}_{\mathsf{A}}^*$ as in Theorem 2. Then,*

1. *the protocol in Fig. 7 is CCA-secure if the underlying key-agreement protocol is secure against active adversaries for Alice and the underlying nonce-based encryption scheme is CCA-secure; and*
2. *$\mathcal{W}_{\mathsf{A}}^*$ preserves CCA-security if the encryption scheme has unique ciphertexts, the key-agreement protocol is authenticated for Bob, and $\mathcal{W}_{\mathsf{A}}$ preserves security for Alice, is exfiltration resistant against Bob with valid transcripts, and detects failure.*

5.1 Key Agreement Secure Against Active Adversaries

Theorem 4 motivates the study of key-agreement protocols with reverse firewalls that preserve security against active adversaries. In the classical setting, the common solution is essentially for each of the parties to sign the transcript of this run of the protocol. However, this solution does not work in our setting because it is important for us that messages *can* be altered without breaking functionality, so that the firewall can rerandomize messages when necessary.

Of course, while we want to allow for the possibility that Alice and Bob see a different transcript but still output a key, we still want them to agree on the key itself. This leads to the idea of signing some deterministic function of the key, so that the signatures can be used to verify that the parties share the same key without necessarily requiring them to share the same transcript. This is the heart of our solution.

We also have to worry that the signatures themselves can provide channels, allowing tampered versions of the parties to leak some information. We solve this by using a unique signature scheme, as defined by [30]. (See [2] for a thorough analysis of signatures in the context of reverse firewalls and corrupted implementations, including alternative ways to implement signatures that would suffice for our purposes.)

Furthermore, in order for our firewall to fail detectably, it has to be able to check the signature itself—so that it can distinguish a valid transcript from an invalid one. So, we would like the parties to sign a deterministic function of g^{ab} that is efficiently computable given only access to g^a and g^b. This leads naturally to the use of a symmetric bilinear map $e : G \times G \to G_T$. The parties then sign $e(g^a, g^b)$. Of course, g^{ab} is no longer indistinguishable from random in the presence of a bilinear map. But, it can be hard to compute. So, we apply a cryptographic hash function H to g^{ab} in order to extract the final key $H(g^{ab})$.

We now provide two definitions to make this precise.

Definition 10 (Unique Signatures). *A* unique signature scheme *is a triple of algorithms* $(\mathsf{KeyGen}, \mathsf{USig}, \mathsf{Ver})$. KeyGen *takes as input* 1^λ *where* λ *is the security parameter and outputs a public key pk and a private key sk.* Sig *takes as input the secret key sk and a plaintext* m *and outputs a signature* τ. Ver *takes as input the public key pk, a signature* τ *and a message* m *and outputs either* true *or* false. *A signature scheme is* correct *if* $\mathsf{Ver}_{pk}(\mathsf{USig}_{sk}(m), m) = \mathsf{true}$. *It is* unique *if for each plaintext* m *and public key pk, there is a unique signature* τ *such that* $\mathsf{Ver}_{pk}(\tau, m) = \mathsf{true}$.

A signature scheme is secure against adaptive chosen-message existential-forgery attacks *if no adversary with access to the public key and a signature oracle can produce a valid signature not returned by the oracle.*

We will need to use a group with a symmetric bilinear map in which the following variant of the computational Diffie-Hellman assumption holds.

Definition 11 (Any-base CDH). *For a group G of order p, we say that* any-base CDH is hard in G *if no probabilistic polynomial-time adversary taking input*

```
Alice                                                                 Bob

INPUT: $sk_\mathsf{A}$            INPUT: $(pk_\mathsf{A}, pk_\mathsf{B}, \mathsf{sid})$            INPUT: $sk_\mathsf{B}$

$a \xleftarrow{\$} \mathbb{Z}_p^*$; $A \leftarrow g^a$
$C \leftarrow \mathsf{Com}(A)$
                                  $\xrightarrow{C}$
                                                                      $b \xleftarrow{\$} \mathbb{Z}_p^*$
                                  $\xleftarrow{B := g^b}$
IF $B = 1_G$,
    OUTPUT $\perp$
$k \leftarrow e(A, B)$
$\tau_\mathsf{A} \leftarrow \mathsf{USig}_{sk_\mathsf{A}}(k, \mathsf{sid})$
                                  $\xrightarrow{(\mathsf{Open}(C), \tau_\mathsf{A})}$
                                                                      IF $A = 1_G$,
                                                                          OUTPUT $\perp$
                                                                      $k \leftarrow e(A, B)$
                                                                      IF NOT $\mathsf{Ver}(\tau_\mathsf{A}, k, \mathsf{sid})$,
                                                                          OUTPUT $\perp$
                                                                      $\tau_\mathsf{B} \leftarrow \mathsf{USig}_{sk_\mathsf{B}}(k, \mathsf{sid})$
                                  $\xleftarrow{\tau_\mathsf{B}}$
IF NOT $\mathsf{Ver}(\tau_\mathsf{B}, k, \mathsf{sid})$,
    OUTPUT $\perp$
OUTPUT $H(B^a)$                                                       OUTPUT $H(A^b)$
```

Fig. 12. Authenticated key agreement with a firewall for both parties. USig is a unique signature.

(g, g^a, g^b) where $g \xleftarrow{\$} G$ and $(a, b) \xleftarrow{\$} \mathbb{Z}_p^2$ has non-negligible probability of returning (h^a, h^b, h^{ab}) for some element $h \in G \setminus \{1_G\}$.

We now present our protocol in Fig. 12 with a reverse firewall for both parties in Fig. 13. It requires a unique signature scheme $(\mathsf{USig}, \mathsf{Ver})$ with public keys pk_A for Alice and pk_B for Bob and corresponding secret keys sk_A and sk_B respectively, a base group G with generator g in which any-base CDH is hard, a target group G_T, and a non-trivial bilinear map between the two groups $e : G \times G \rightarrow G_T$. We also need a function $H : G \rightarrow \{0, 1\}^\ell$ for some polynomially large ℓ that extracts hardcore bits from CDH. Presumably a standard cryptographic hash function will work. For simplicity, we model H as a random oracle, but we note that the proof can be modified to apply to any function H such that $(g^a, g^b, H(g^{ab}))$ is indistinguishable from random. (See [27] for a discussion of such functions.) We stress again that this protocol is remarkably efficient, and we think that it can

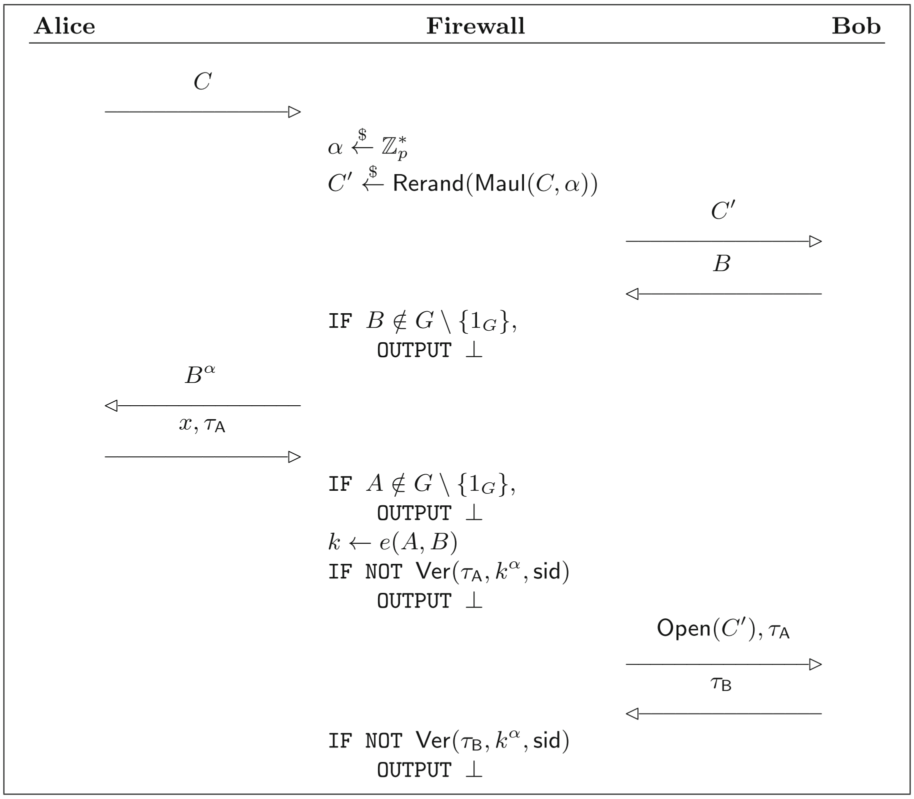
Alice
Firewall
Bob
C
$\alpha \xleftarrow{\$} \mathbb{Z}_p^*$
$C' \xleftarrow{\$} \mathsf{Rerand}(\mathsf{Maul}(C, \alpha))$
C'
B
IF $B \notin G \setminus \{1_G\}$,
OUTPUT $\perp$
B^α
x, τ_{A}
IF $A \notin G \setminus \{1_G\}$,
OUTPUT $\perp$
$k \leftarrow e(A, B)$
IF NOT $\mathsf{Ver}(\tau_{\mathsf{A}}, k^\alpha, \mathsf{sid})$
OUTPUT $\perp$
$\mathsf{Open}(C'), \tau_{\mathsf{A}}$
τ_{B}
IF NOT $\mathsf{Ver}(\tau_{\mathsf{B}}, k^\alpha, \mathsf{sid})$
OUTPUT $\perp$

Fig. 13. Reverse firewall for either Alice or Bob in the protocol from Fig. 12. C is a commitment of the group element A. $\mathsf{Maul}(C, \alpha)$ takes a commitment $C = \mathsf{Com}(A)$ and converts it into a commitment of A^α. $\mathsf{Rerand}(C)$ takes a commitment $C = \mathsf{Com}(A)$ and converts it into a uniformly random commitment of A. We assume that a rerandomized and mauled commitment can be opened with access to an opening of the original commitment.

and should be used in practice. The proof of the following theorem is in the full version [22].

Theorem 5. *The protocol shown in Fig. 12 is authenticated for Bob and secure against active adversaries if the signature scheme is secure and any-base CDH is hard in G. The reverse firewall $\mathcal{W}$ shown in Fig. 13 preserves security against active adversaries for Alice, preserves authenticity, is exfiltration resistant for Alice against Bob with valid transcripts, and detects failure for Alice. $\mathcal{W}$ also preserves security against active adversaries for Bob, is exfiltration resistant for Bob against Alice with valid transcripts, and detects failure for Bob.*

References

1. Alwen, J., Shelat, A., Visconti, I.: Collusion-free protocols in the mediated model. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 497–514. Springer, Heidelberg (2008)
2. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: CCS (2015)
3. Ball, J., Borger, J., Greenwald, G.: Revealed: how US and UK spy agencies defeat internet privacy and security. Guardian Weekly, September 2013
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 26. Springer, Heidelberg (1998)
5. Bellare, M., Hoang, V.T.: Resisting randomness subversion: fast deterministic and hedged public-key encryption in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 627–656. Springer, Heidelberg (2015)
6. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: strongly undetectable algorithm-substitution attacks. In: CCS (2015)
7. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). Full version: [8]
8. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. Cryptology ePrint Archive, report 2014/438 (2014). http://eprint.iacr.org/
9. Blaze, M., Bleumer, G., Strauss, M.J.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
10. Burmester, M., Desmedt, Y.G.: All Languages in NP have divertible zero-knowledge proofs and arguments under cryptographic assumptions. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 1–10. Springer, Heidelberg (1991)
11. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
12. Vulnerability summary for CVE-2014-1260 (‘Heartbleed’), April 2014. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1260
13. Vulnerability summary for CVE-2014-1266 (‘goto fail’), February 2014. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1266
14. Vulnerability summary for CVE-2014-6271 (‘Shellshock’), September 2014. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
15. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 579–598. Springer, Heidelberg (2015)
16. Desmedt, Y.G.: Abuses in cryptography and how to fight them. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 375–389. Springer, Heidelberg (1990)
17. Desmedt, Y.: Subliminal-free sharing schemes. In: Information Theory (1994)
18. Desmedt, Y.G., Pieprzyk, J., Steinfeld, R., Wang, H.: A non-malleable group key exchange protocol robust against active insiders. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 459–475. Springer, Heidelberg (2006)

19. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)
20. Dodis, Y., Fiore, D.: Interactive encryption and message authentication. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 494–513. Springer, Heidelberg (2014)
21. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A formal treatment of backdoored pseudorandom generators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 101–126. Springer, Heidelberg (2015)
22. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls–secure communication on corrupted machines. Cryptology ePrint Archive, report 2015/548 (2015). http://eprint.iacr.org/2015/548
23. Greenwald, G.: No Place to Hide: Edward Snowden the N.S.A. and the U.S. Surveillance State. Metropolitan Books, New York (2014)
24. Groth, J.: Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 152–170. Springer, Heidelberg (2004)
25. Juniper vulnerability (2015). https://kb.juniper.net/InfoCenter/index?page=content&id=JSA10713
26. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: CCS (2005)
27. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on Gap Hashed Diffie-Hellman. In: PKC (2007)
28. Lenstra, A.K., Hughes, J.P., Augier, M., Bos, J.W., Kleinjung, T., Wachter, C.: Public keys. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 626–642. Springer, Heidelberg (2012)
29. Lepinksi, M., Micali, S., Shelat, A.: Collusion-free protocols. In: STOC (2005)
30. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: FOCS (1999)
31. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 657–686. Springer, Heidelberg (2015)
32. Okamoto, T., Ohta, K.: Divertible zero knowledge interactive proofs and commutative random self-reducibility. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 134–149. Springer, Heidelberg (1990)
33. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
34. Perlroth, N., Larson, J., Shane, S.: National Security Agency able to foil basic safeguards of privacy on Web. The New York Times, September 2013
35. Pieprzyk, J., Wang, H.: Key control in multi-party key agreement protocols. In: Workshop on Coding, Cryptography and Combinatorics (2003)
36. Prabhakaran, M., Rosulek, M.: Rerandomizable RCCA encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 517–534. Springer, Heidelberg (2007)
37. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Cliptography: clipping the power of kleptographic attacks. Cryptology ePrint Archive, report 2015/695 (2015). https://eprint.iacr.org/2015/695
38. Schneier, B., Fredrikson, M., Kohno, T., Ristenpart, T.: Surreptitiously weakening cryptographic systems. Technical report, IACR Cryptology ePrint Archive, 2015: 97 (2015). http://eprint.iacr.org/2015/97
39. Simmons, G.: The prisoners' problem and the subliminal channel. In: Chaum, D. (ed.) CRYPTO 1983, pp. 51–67. Springer, New York (1984)
40. https://www.us-cert.gov/ncas/alerts/TA15-051A. February 2015

A The symmetric-key setting

Here, we consider the setting in which Alice and Bob share a private key. We observe that a one-round protocol due to Bellare, Paterson, and Rogaway provides a solution that does not even need a reverse firewall [7]. We also use this scheme elsewhere to build protocols that do not rely on shared private keys. We first define nonce-based encryption.

Definition 12. (Nonce-based encryption). *A* nonce-based symmetric-key encryption scheme *is a pair of deterministic algorithms* $(\mathsf{Enc}, \mathsf{Dec})$. Enc *takes as input a key from a key space* $\mathcal{K}$*, a nonce from a nonce space* $\mathcal{N}$*, and a plaintext from a plaintext space* $\mathcal{M}$ *and outputs a ciphertext from a ciphertext space* $\mathcal{C}$*.* Dec *takes as input a key, a nonce, and a ciphertext and returns a plaintext or the special symbol* $\perp$*. The scheme is* correct *if for any key sk, nonce r, and plaintext m,* $\mathsf{Dec}(r, \mathsf{Enc}_{sk}(r, m)) = m$*.*

Such a scheme is CPA secure *if no probabilistic polynomial-time adversary can distinguish between* $\mathsf{Enc}_{sk}(r^*, m_0)$ *and* $\mathsf{Enc}_{sk}(r^*, m_1)$ *with non-negligible advantage where* r^**,* m_0*, and* m_1 *are adversarially chosen when given access to an encryption oracle that outputs* $\perp$ *unless given a unique nonce r. It is* CCA-secure *if no probabilistic polynomial-time has non-negligible advantage when also given access to a decryption oracle that outputs* $\perp$ *if* $r = r^*$*.*

Such a scheme $(\mathsf{Enc}, \mathsf{Dec})$ has unique ciphertexts *if for any key sk, message m, and nonce r, there is exactly one ciphertext c such that* $\mathsf{Dec}(r, c) = m$*.*

Theorem 6. *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a nonce-based symmetric-key encryption scheme. Then, if the encryption scheme is CPA-secure (resp. CCA-secure) the one-round protocol in which Alice sends* $\mathsf{Enc}_{sk}(\mathsf{sid}, m)$ *and Bob returns* $\mathsf{Dec}_{sk}(\mathsf{sid}, m)$ *is a CPA-secure (resp. CCA-secure) one-round privately keyed message-transmission protocol without forward secrecy. If the encryption scheme has unique ciphertexts, then the "trivial" reverse firewall that simply passes Alice's messages to Bob unchanged preserves security and is exfiltration resistant against Bob.*

See, e.g., [7] for formal analysis and the construction of such a scheme. The key thing to note from our perspective is that, as Bellare et al. observe, the fact that the encryption scheme has unique ciphertexts implies that any tampered version of Alice that maintains functionality necessarily behaves identically to honest Alice. The next theorem shows that we essentially cannot do any better for a one-round protocol without using public-key primitives. The proof is in the full version.

Theorem 7. *There is a black-box reduction from semantically secure public-key encryption to CPA-secure symmetric-key encryption with at least four possible plaintexts and a reverse firewall that strongly preserves CPA security.*

Of course, the primary drawbacks of this approach are that it requires Alice and Bob to share a secret key and that it does not offer forward secrecy.

B Less efficient one-round protocols

Finally, we show one-round protocols in the singly keyed setting, in which Bob has a public-key/private-key pair, but Alice does not. These are essentially the single-round analogue of the two-round protocol presented in Fig. 4 in Sect. 3. They can also be thought of as the natural extension of public-key encryption schemes to the reverse firewall setting. (In particular, these protocols are not efficient, in the sense that they use public-key operations on the plaintext, which may be very long.) Indeed, we show that the existence of such a protocol is equivalent to the existence of rerandomizable encryption, and we show how to achieve CCA-security (though not forward secrecy).

B.1 One-round CPA-secure protocols

The next theorem shows that one-round CPA-secure protocols with reverse firewalls are equivalent to rerandomizable public-key encryption. The proof is in the full version [22].

Theorem 8. *Any (strongly) rerandomizable semantically secure public-key encryption scheme implies a one-round CPA-secure singly keyed message-transmission protocol without forward security with a reverse firewall that (strongly) preserves security and (strongly) resists exfiltration. Conversely, any one-round CPA-secure message-transmission protocol with a reverse firewall that (strongly) preserves security implies a (strongly) rerandomizable semantically secure public-key encryption scheme.*

B.2 A one-round CCA-secure protocol

To extend this idea to stronger notions of security, we need the underlying encryption scheme to satisfy stronger notions of security. A natural candidate is CCA security. However, CCA-secure encryption schemes cannot be rerandomizable, so we need a slightly weaker notion. RCCA security, as defined by [11], suffices, and rerandomizable RCCA-secure schemes do exist (see, e.g., [24,36]), though they are relatively inefficient. (They are not *strongly* rerandomizable; their rerandomization procedures do not work on invalid ciphertexts.) We present the RCCA security game in Fig. 14. In addition, we need a rerandomized ciphertext to be indistinguishable from a valid encryption *even with access to a decryption oracle.* Figure 15 and the definition below makes this precise.

Definition 13. *An encryption scheme is* RCCA secure *if no probabilistic polynomial-time adversary $\mathcal{E}$ has non-negligible advantage in the game presented in Fig. 14. It is* RCCA rerandomizable *if there exists an algorithm* Rerand *with access to the public key such that for any ciphertext c with* $\mathsf{Dec}(c) \neq \perp$, $\mathsf{Dec}(\mathsf{Rerand}(c)) = \mathsf{Dec}(c)$ *and no probabilistic polynomial-time adversary $\mathcal{E}$ has non-negligible advantage in the game presented in Fig. 15 when we require that* $\mathsf{Dec}(c_i) \neq \perp$. *It is* strongly RCCA-rerandomizable *if the previous statement holds even if* $\mathsf{Dec}(c_i) = \perp$.

```
proc. IND-RCCA($\lambda$)
$(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{E}^{\mathcal{O}_1}(pk)$
$b \xleftarrow{\$} \{0,1\}$; $C^* \xleftarrow{\$} \mathsf{Enc}_{pk}(m_b)$
$b^* \leftarrow \mathcal{E}^{\mathcal{O}_2}(\sigma, c^*)$
OUTPUT $(b = b^*)$

proc. $\mathcal{O}_1(c)$
OUTPUT $\mathsf{Dec}_{sk}(c)$

proc. $\mathcal{O}_2(c)$
$m \leftarrow \mathsf{Dec}_{sk}(c)$
IF $m = m_0$ OR $m = m_1$,
    OUTPUT Challenge
ELSE,
    OUTPUT $m$
```

Fig. 14. The RCCA security game.

```
proc. IND-RCCA($\lambda$)
$(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(c_0, c_1) \leftarrow \mathcal{E}^{\mathcal{O}_1}(pk)$
$b \xleftarrow{\$} \{0,1\}$; $c^* \xleftarrow{\$} \mathsf{Rerand}_{pk}(c_b)$
$b^* \leftarrow \mathcal{E}^{\mathcal{O}_2}(c^*)$
OUTPUT $(b = b^*)$

proc. $\mathcal{O}_1(c)$
OUTPUT $\mathsf{Dec}_{sk}(c)$

proc. $\mathcal{O}_2(c)$
$m \leftarrow \mathsf{Dec}_{sk}(c)$
IF $m = \mathsf{Dec}_{sk}(c_0)$ OR $m = \mathsf{Dec}_{sk}(c_1)$,
    OUTPUT Challenge
ELSE,
    OUTPUT $m$
```

Fig. 15. The RCCA rerandomization game.

The below theorem is the CCA analogue of Theorem 8. The proof is in the full version [22].

Theorem 9. *Any (strongly) RCCA-rerandomizable, RCCA-secure encryption scheme implies a one-round CCA-secure singly keyed message-transmission protocol without forward security with a reverse firewall that (strongly) preserves security and (strongly) resists exfiltration.*