UC-Secure Cryptographic Reverse Firewall–Guarding Corrupted Systems with the Minimum Trusted Module

Geng Li[1], Jianwei Liu[2], Zongyang Zhang[2,⋆], and Yanting Zhang[2]

[1] National Computer Network Emergency Response Technical Team/Coordination Center of China, China
[2] School of Cyber Science and Technology, Beihang University, China
{ligeng,liujianwei,zongyangzhang,yantingzhang}@buaa.edu.cn

Abstract. Nowadays, mass-surveillance is becoming an increasingly severe threat to the public's privacy. The PRISM and a series of other events showed that inner attacks such as subversion attacks may exist in the current network extensively. As an important strategy to defend users' privacy against these attacks, cryptographic reverse firewall (CRF) is designed to be a middle-box, modifying all the messages coming in and out of a computer. However, the current formal definition of CRFs merely considers a single protocol session. If such a CRF applies to multiple entities, the security of every entity could not be deduced directly, which leads to an application limitation. In this work, we re-define the notion of CRF from a new perspective based on UC-emulation. Our new definition expresses all expected properties of a CRF in a more brief way, under the universal composition environment. We present a composition theorem which enables deploying one CRF for a local area of network rather than a single computer, and this can significantly reduce the number of CRFs used in practical applications. As another part of this work, under the new definition, we present the first deterministic CRF construction. Compared with existing CRFs, our construction only requires secure randomness in its initial phase rather than every protocol session, and such randomness can be acquired from a public resource. Noting that the probabilistic algorithms are the main targets of subversion attacks, our work makes it much easier to realize a trusted CRF, and thus, pushes CRFs from a concept to application with one more step.

Keywords: Post-Snowden cryptography · Subversion attack · Cryptographic reverse firewall · Universal composition · Security implementation

1 Introduction

The PRISM has attracted great attention to the risk of mass-surveillance. A series of events implied that some powerful institutions may monitor people's private conversation and other online activities, via some special methods which are out of the consideration of conventional cryptography. Subversion attack is a typical attack method in the mass-surveillance, which deviates the implementations of cryptographic algorithms from the corresponding specifications in an undetectable way, and only the backdoor holder is able to recognize the subverted implementations and acquire user's private information based on their outputs. Due to the effectiveness and concealment, the subversion attack has been explored by a lot of works, and it serves as an essential foundation for Post-Snowden cryptography.

To deal with subversion attacks, researchers have proposed several defending strategies, such as the split-program [17], multi-source components [13], public random resources [1], and cryptographic reverse firewalls (CRFs) [15,10,8]. CRF is a machine sitting between a user's computer and the outside world, modifying all the incoming and outgoing messages to guard the protocol against information leakage. On the one hand, a CRF is assumed to be a trusted module, i.e., the implementation should completely obey its specification. On the other hand, however, a CRF is still a public entity, saying, it can only access public information such as public-keys and ciphertexts. Mironov and Stephens-Davidowitz [15] formally defined a CRF with three important properties: functionality-maintaining, security-preserving, and exfiltration-resistance.

⋆ Corresponding author

Assumed to be trusted, a CRF seems to exclude the risk of subversion attacks in an ideal way. However, the current theory is a little imperfect from the view of practical applications. Following the definition in work [15], we need to deploy a CRF for every entity in the reality (as shown in Fig. 1). This may imply a logical contradiction: we worry about the credibility of user's computer, but at the same time, we directly assume the existence of the same amount of trusted modules (CRF). Therefore, the idea of CRF just bypasses the difficulty in the mass-surveillance, and the tricky problem remains unsolved.

We claim that in a reasonable design, the number of CRFs should be much less than the number of users' computers, thus, we can concentrate to achieving these small amount of trusted modules to protect the whole system. Concretely, as shown in Fig. 2, a CRF serves for a local area of network containing multiple computers. It is straightforward that this framework is much more practical than the previous "one-one" pattern. However, this "one-many" framework is actually out of the consideration of the current definition. Here, several entities share a CRF, and a CRF may serve for multiple kinds of protocol. It is inevitable that different protocol sessions may have shared information or joint state. We cannot ensure that a CRF under the current definition is able to protect every entity in such a compositional setting. Thus, we call for a new definition of CRFs to adapt to this setting.

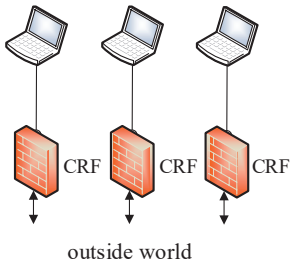

Fig. 1. One-one pattern of CRF deployment

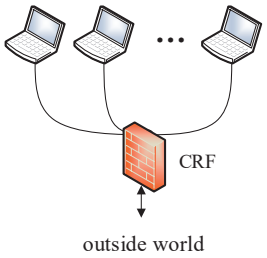

Fig. 2. One-many pattern of CRF deployment

Besides, another factor that inspires us to revisit the formalization of CRF is the redundancy of the current definition [15], where the relationship between security-preserving and exfiltration-resistance is much complex. These two properties sometimes are equivalent, but sometimes have subtle differences. Including both of them as basic requirements of a CRF seems redundant, while omitting either one will render the definition incomplete. Therefore, it is meaningful if we could find a method to conclude all the expected properties of a CRF with only one expression.

Under the current definition [15], researchers now have proposed a series of constructions of CRF. To the best of our knowledge, all these CRFs need to generate the same amount of randomness to the underlying protocol. We should note that achieving the credibility of probabilistic implementations is far more difficult than deterministic implementations, thus, generating such a large amount of randomness increases the difficulty of realizing a trusted CRF. If we could construct a CRF with only deterministic algorithms, the meaning of CRFs will be significantly increased. This work may also initialize another line of research, i.e., removing the assumption that CRFs are trusted modules, and including CRFs also under the consideration of subversion attacks.

1.1 Our Contribution

The main purpose of this paper about CRF is "guarding corrupted systems with the minimum trusted module", and this keynote can be interpreted from two levels. In the first level, we propose the scenario that using one CRF to guard multiple computers. We re-define the notion of CRF based on the UC framework,

which is more compatible with such compositional settings. In the second level, we try to further shrink the trusted element of CRF, by designing a CRF only using deterministic algorithms.

Re-definition of a CRF based on UC-emulation. Based on the UC-emulation[5,6], we re-define the notion of CRF. All analyses in this paper are under the precondition that the specification of a protocol can emulate the ideal functionality perfectly. We say a CRF guards the protocol, if either it could detect the subverted implementations, or it could correct the corrupted protocol to emulate the ideal functionality, with no environment being able to distinguish between these two cases. We build the new definition by three steps: Firstly, we specify the formalization of a protocol using the language of UC framework; Then, we formulate the deployment and operation of a CRF; Finally, we present the security definition of a CRF.

Compared with the current one, our new definition has at least two advantages. Firstly, this definition is designed from the view of composition. We present a composition theorem, which demonstrates that the CRF's security property for every single session does not decline in any compositional setting. That is to say, when a CRF deals with multiple sessions or multiple protocols at the same time, even though different sessions have shared information or joint state, every protocol session can still UC-emulate the ideal functionality. Thus, such definition provides a basis for the "one-many" deployment in applications as shown in Fig. 2. Secondly, the new definition incorporates the three basic requirements (functionality-maintaining, security-preserving and exfiltration-resistance) into one property, which makes the expression far more clear. We make a rigorous comparison between the two definitions of a CRF, proving that the new definition implies the current one in a general sense.

Construction of a deterministic CRF. Our construction is based on the classical framework of CRF in work [10], but transfers the probabilistic algorithms to deterministic algorithms. Noting that probabilistic algorithms are main targets of subversion attacks, this modification will significantly weaken the assumption that a CRF is a trusted module.

Unlike the existing CRFs which need to generate randomness during every run of protocol, our CRF only requires a small amount of randomness as its keys in the initial phase. The keys should remain confidential to the protocol parties, however, may be a little surprising, they are unnecessary to stay secure against external adversaries. Such property provides us with a possible operation pattern of CRF, i.e., injecting a public randomness into a CRF, after the implementations of protocol parties are deployed.

We achieve such a deterministic CRF by replacing the coins in the former CRFs, via hashing input messages along with the CRF key. Observing that even if being subverted, an implementation of protocol party should still ensure its output to have high min-entropy, otherwise it can be detected from its specification easily. Taking use of this point, we prove that for both subverted implementations and external adversaries, the input of the hash function is inaccessible, thus, the acquired coins can serve as uniform randomness.

1.2 Related Work

Post-Snowden Cryptography. The notion of kleptography was introduced by Young and Yung in a series of researches [18,19,20,21]. They considered the scenario where implementations of cryptographic algorithms are maliciously designed by adversaries. They also designed several detailed attacks aimed at some commonly used schemes such as RSA, Elgamal, Diffie-Hellman key-exchange protocol, et al.

Bellare, Paterson and Rogaway [4] formally established the security model for encryption schemes under subversion attacks. This model consists of a pair of games, the surveillance game and the detection game, to characterize the advantage of a surveillant and the probability of detecting a subverted implementation. Work [9] and work [3] made partial progresses on the model in work [4], to adopt to several special cases such as input-trigger attacks. Using a similar framework, Ateniese, Magri and Venturi [2] formulated the security model for signature schemes. Russell et al. [16] proposed the notion of cliptography. They introduced an entity named "watchdog" which checks all implementations provided by an adversary. If all implementations agreed by the watchdog could run without any difference to the specification even from the view of the adversary itself, then we say the scheme is stego-freeness.

A series of defending strategies have been proposed against the subversion attack. Russell et al. [17] gave the idea of split-program methodology, dividing one algorithm into several components. All implementations with respect to the components are probably subverted and modeled as blackboxes, but users could ensure

the security of the whole system with the help of the watchdog's checking and a trusted amalgamation. Fischlin and Mazaheri [11] presented a defending strategy named "self-guarding protocol". They assumed that users can set up a secure initial phase for the protocol, in which a number of randomness are generated, and all algorithms executed after the initial phase are designed to be deterministic. Ateniese et al. [1] showed how to correct subverted implementations using a public (secure) randomness generator which is accessible to both users and adversaries. Li, Liu and Zhang [13,14] proposed a defending strategy where users could construct a system using implementations from multiple sources. When multiple adversaries are isolated [13] or only able to communicate in a limited way [14], users can take certain designs to achieve a secure system against all adversaries, only using untrusted implementations.

Cryptographic Reverse Firewall. The notion of cryptographic reverse firewall (CRF) was first proposed by Mironov and Stephens-Davidowitz [15]. They characterized a robust CRF by three properties: functionality-maintaining, security-preserving and exfiltration-resistance. To demonstrate the achievability of this definition, they also showed how to convert an arbitrary protocol into a protocol with exfiltration-resistant reverse firewalls for all parties.

Dodis, Mironov and Stephens-Davidowitz [10] researched the security of message-transmission protocols under subversion attacks. They designed a CRF for a type of interactive and concurrent CCA-secure message-transmission. Chen et al. [8] extended the notion of smooth projective hash function (SPHF) to malleable smooth projective hash function, based on which they proposed a general construction of CRF for some widely used cryptographic protocols. Especially, considering conventional oblivious transfer (OT) protocol is not compatible with the above modular way of CRF construction, they developed a new OT framework from graded rings and showed how to construct OT-CRFs by modifying the malleable SPHF framework.

Universal Composition. First proposed by Canetti [5], universal composition is a general framework to describe cryptographic protocols and analyze their security in complex environments. Canetti defined the security of a protocol by the indifference between an execution of the real protocol with an adversary, and an execution of the ideal functionality with a PPT simulator. Within this framework, protocols are guaranteed to maintain their security in any context, even in the presence of an unbounded number of arbitrary protocol sessions that run concurrently in an adversarially controlled manner.

The composition theorem in work [5] assumed the composed protocol instances have disjoint internal states and are independent completely. To relax this restriction, Canetti and Rabin [7] extended the framework and proposed the universal composition with joint state (JUC). This work considered the case where different protocol instances may have joint state and randomness. Following the line of work [7], Canetti et al. [6] further explored the composition theorem in the scenario where we cannot ensure that the set-up phase of a protocol provides the expected security guarantee. Hofheinz and Shoup [12] proposed a new framework named GNUC, which deviate from UC in several important aspects such as the formalization of protocols and the notion of corruptions.

Organization. Section 2 introduces the main notations and preliminaries. Section 3 presents a formal description of protocols, and gives a definition of CRFs from a new prospective. Section 4 presents the composition theorem of protocol instances equipped with a CRF. Section 5 demonstrates that the new definition implies the current one. Section 6 presents a construction of deterministic CRF under the new definition.

2 Preliminary

2.1 Notations

We use $s \xleftarrow{\$} \mathcal{S}$ to denote that s is a uniformly random element in set $\mathcal{S}$. U_ℓ denotes a ℓ-bit uniformly random string. $[1, r]$ is short for set $\{1, 2, \cdots, r\}$. $s_1||s_2$ means the concatenation of two bit strings s_1 and s_2. $\mathsf{poly}(x)$ represents a polynomial function of x. Let λ be a security parameter. $\mathsf{negl}(\lambda)$ is a negligible function in λ if it vanishes faster than the inverse of any polynomial in λ. $\mathsf{A}(x)$ is a probabilistic polynomial-time (PPT) algorithm if for any input x, $\mathsf{A}(x)$ terminates at most $\mathsf{poly}(|x|)$ steps. If the algorithm A outputs y upon the input x, we write $y \leftarrow \mathsf{A}(x)$. We use $\mathsf{G} \twoheadleftarrow \mathsf{A}(\cdot)$ to represent that A generates the implementation G.

For two distributions $\mathcal{X} = \{X_\lambda\}$ and $\mathcal{Y} = \{Y_\lambda\}$, let $\mathsf{Dist}_{\mathsf{D}}(\mathcal{X}, \mathcal{Y}) \stackrel{def}{=} \big|\Pr[\mathsf{D}(X_\lambda) = 1] - \Pr[\mathsf{D}(Y_\lambda) = 1]\big|$. If $\mathsf{Dist}_{\mathsf{D}}(\mathcal{X}, \mathcal{Y}) \leq \mathsf{negl}(\lambda)$ for all PPT algorithm D, we call $\mathcal{X}$ and $\mathcal{Y}$ are computational indistinguishable and denote it by $\mathcal{X} \approx \mathcal{Y}$ for simplicity.

2.2 Proud-but-Malicious Adversary

"Proud-but-malicious" adversaries are the main issue considered in Post-Snowden cryptography. This kind of adversary tries to compromise security by providing subverted implementations of cryptographic algorithms. The important thing is that, a "proud-but-malicious" adversary prefers subversions to be "under the radar" of any possible detection [17]. Russell et al. established the notion of watchdog $\mathcal{W}$ which is a PPT detector aiming at distinguishing subversion from the specification. Concretely, for $\mathsf{G}_{\mathsf{impl}} \twoheadleftarrow \mathcal{A}(\mathsf{G}_{\mathsf{spec}})$, and any PPT watchdog $\mathcal{W}$, we call $\mathcal{A}$ is a proud-but-malicious adversary against offline watchdog if

$$\big|\Pr[\mathcal{W}^{\mathsf{G}_{\mathsf{impl}}}(1^\lambda) = 1] - \Pr[\mathcal{W}^{\mathsf{G}_{\mathsf{spec}}}(1^\lambda) = 1]\big| \leq \mathsf{negl}(\lambda).$$

Similarly, we call $\mathcal{A}$ is a proud-but-malicious adversary against online watchdog if

$$\big|\Pr[\mathcal{W}^{\mathsf{G}_{\mathsf{impl}}}(\tau) = 1] - \Pr[\mathcal{W}^{\mathsf{G}_{\mathsf{spec}}}(\tau) = 1]\big| \leq \mathsf{negl}(\lambda),$$

where τ denotes all the transcripts accessible to $\mathcal{A}$ in the same time. When $\mathcal{W}$ is clear, we simply say $\mathcal{A}$ is a proud-but-malicious adversary. And for simplicity, in this paper we call it a *PM-adversary.*

2.3 The Current Definition of a CRF

The notion of cryptographic reverse firewall (CRF) was first proposed by Mironov and Stephens-Davidowitz [15]. A CRF is a trusted module sitting between a user's computer and the outside world, intercepting all the messages coming in and out. A formal description of CRF is presented in Definition 1

Definition 1. *A cryptographic reverse firewall (CRF) is a stateful algorithm* crf *that takes as input its state and a message, and outputs an updated state and a modified message. For simplicity, we do not write the state of* crf *explicitly.*

For a cryptographic reverse firewall crf *and a party* $\mathsf{A} = \{\mathsf{receive}, \mathsf{next}, \mathsf{output}\}$*, where* $\mathsf{receive}$ *is the function to receive messages from outside,* next *is the function to generate outputting message, and* output *is the function to generate the final result of the protocol. The composed party* $\mathsf{A} \circ \mathsf{crf}$ *is defined as:*

$$\mathsf{A} \circ \mathsf{crf} \stackrel{def}{=} \Big(\mathsf{receive}_{\mathsf{A}\circ\mathsf{crf}}(\sigma, m) = \mathsf{receive}_{\mathsf{A}}(\sigma, \mathsf{crf}(m)), \mathsf{next}_{\mathsf{A}\circ\mathsf{crf}}(\sigma) = \mathsf{crf}(\mathsf{next}_{\mathsf{A}}(\sigma)), \mathsf{output}_{\mathsf{A}\circ\mathsf{crf}}(\sigma) = \mathsf{output}_{\mathsf{A}}(\sigma)\Big),$$

where σ *denotes the state of the party* A*, and* m *denotes the transcript received.*

According to work [15], a CRF is characterized by three properties: functionality-maintaining, security-preserving, and exfiltration-resistance. Readers are referred to Appendix A for the formal definitions.

3 Security Model

Based on the notion of UC-emulation, this section presents a new definition of CRFs, which is totally different from the current definition consisting of functionality-maintaining, security-preserving and exfiltration-resistance. We first make a formal characterization of protocols in the view of the subversion attacks in Section 3.1. Next, we state the deployment and operation of a CRF in Section 3.2. Based on the above works, we present the security definition of a CRF in Section 3.3.

3.1 Protocol Framework

Our characterization of protocols mainly follows the idea in work [5], but with a deep simplification. We simplify the model mainly because the intention of this paper concentrates on CRFs, rather than protocols themselves. Thus, it is unnecessary to analyze a protocol in such a complicated way. The subversion attack could be regarded as "inner attack", i.e., an adversary compromises the security by corrupting the implementations inside a user's system. Considering such a property, we divide a protocol into several inner parts (main machines) and an external part (channel machine).

We consider an n-party protocol $\mathcal{P}$, an adversary $\mathcal{A}$ and an environment $\mathcal{E}$, all of which are modeled as interactive Turing machines (ITMs). Each instance of ITM is labeled by a protocol identity fid, a party identity pid and a session identity sid, which remain unchanged throughout. The messages transmitted between instances of ITM are interpreted as a tuple:

$$(fid, sid, pid_s, pid_r, m||ad, type).$$

pid_s and pid_r are identities of the sender and the receiver, respectively. $m||ad$ is the message and its associated data. $type \in \{$"*input*", "*internal-output*", "*output*", "*backdoor*"$\}$ is the label of the message; "*input*" represents the initial input of the protocol from environment $\mathcal{E}$; "*internal-output*" represents the information transmitted among machines of protocol $\mathcal{P}$; "*output*" represents the final output of protocol $\mathcal{P}$ from party π_i to $\mathcal{E}$; "*backdoor*" is used to represent backdoor information come from the adversary, or disclosed to it.

We simplify the structure of a protocol in work [5] and define $\mathcal{P}$ only as a channel machine and several main machines: $\mathcal{P} = \{\pi_1, \cdots, \pi_n, \mathsf{ch}\}$. Formally,

- π_i ($i \in [1, n]$) represents the program executed by each party in the protocol. π_i receives information from $\mathcal{E}$ in type of "*input*", and returns information in type of "*output*". π_i also has interface with ch to transmit messages in type of "*internal-output*", and has interface with $\mathcal{A}$ to transmit messages in type of "*backdoor*". Especially, there is no direct interface between π_i and π_j ($i, j \in [1, n]$). All their communication is via channel machine or adversaries.
- ch represents the message transmission during the execution of the protocol. Besides interacting with π_i, it is also accessible to the adversary $\mathcal{A}$. For example, for an authenticated channel, when ch receives $(fid, sid, pid_{\pi_i}, pid_{\mathsf{ch}}, m||$"$pid_{\pi_i}$ *to* pid_{π_j}", "*internal-output*"$)$, it sends $(fid, sid, pid_{\mathsf{ch}}, pid_{\mathcal{A}}, m||$"$pid_{\pi_i}$ *to* pid_{π_j}", "*backdoor*"$)$ to $\mathcal{A}$. After receiving $(fid, sid, pid_{\mathcal{A}}, pid_{\mathsf{ch}}$, "*ok*", "*backdoor*"$)$ from $\mathcal{A}$, it sends $(fid, sid, pid_{\mathsf{ch}}, pid_{\pi_j}, m||$"$pid_{\pi_i}$ *to* pid_{π_j}", "*internal-output*"$)$ to party π_j.

3.2 Protocol Equipped with a CRF

Now we present the formalization of a cryptographic reverse firewall. A CRF is an interactive Turing machine which intercepts and modifies all the messages sent between main machines π_i and channel machine ch. Especially, if the protocol is executed normally, the CRF seems "transparent". A CRF is regarded as a trusted model, saying, different with the protocol $\mathcal{P}$, it should be executed honestly according to its specification in any case. Another important difference between the CRF and machines of $\mathcal{P}$ is that a CRF instance is designed to operate across multiple sessions or even multiple protocols, while every ITM instance of protocol is corresponding to only one session.

Credibility and session-units. The adversary is not allowed to corrupt crf. All the messages transmitted between π_i and crf are inaccessible to the adversary. As a CRF operates across multiple sessions, we interpret crf as a basic-program and several session-units. Every session-unit is marked by a triple (fid, sid, pid), where fid and sid are protocol identity and session identity, and pid denotes the protocol party this unit serves. The session-units store the relative information with respect to the corresponding session and party.

The authority to alarm. We allow CRFs to "alarm". If a CRF perceives any abnormality of inputting messages implying that the implementation of the protocol is likely to be subverted, the CRF is able to alarm directly, and the whole protocol breaks off at the same time. To ensure the reasonability, the current definition (see Appendix A) imposes an adversary with some special limitations, e.g., it must provide functionality-maintaining implementations. Our design of "alarm" provides a possibility to remove these limitations and solve the problem by CRF itself.

The notion of shell machine. To capture the feature of transparency of crf, we introduce the notion of shell machine to characterize a protocol deployed with a CRF. A shell machine operates like a "shell", modifying the string fields of the identities of sender and receiver, when transmitting messages coming in or out of the inner machine. We denote the shell of π_i, ch and crf by SH_{π_i}, $\mathsf{SH}_{\mathsf{ch}}$ and $\mathsf{SH}_{\mathsf{crf}}$, respectively, and denote the shelled machine of X by X^{SH}. The shell machines operate as in Fig. 3. Formally:

- SH_{π_i} filters all the messages coming in/out of π_i. It changes outgoing messages $(\cdot, \cdot, pid_{\pi_i}, pid_{\mathsf{ch}}, \cdot, \cdot)$ to $(\cdot, \cdot, pid_{\pi_i}, pid_{\mathsf{crf}}, \cdot, \cdot)$, and forwards other messages without modification.
- $\mathsf{SH}_{\mathsf{ch}}$ filters all the messages coming in/out of ch. It changes outgoing messages $(\cdot, \cdot, pid_{\mathsf{ch}}, pid_{\pi_i}, m||ad, \cdot)$ to $(\cdot, \cdot, pid_{\mathsf{ch}}, pid_{\mathsf{crf}}, m||ad||\text{"}to\ pid_{\pi_i}\text{"}, \cdot)$, and forwards other messages without modification.
- $\mathsf{SH}_{\mathsf{crf}}$ filters all the messages coming in/out of crf. It changes outgoing messages $(\cdot, \cdot, pid_{\mathsf{crf}}, pid_{\pi_i}, \cdot, \cdot)$ to $(\cdot, \cdot, pid_{\mathsf{ch}}, pid_{\pi_i}, \cdot, \cdot)$, and changes outgoing messages $(\cdot, \cdot, pid_{\mathsf{crf}}, pid_{\mathsf{ch}}, \cdot, \cdot)$ corresponding to the session-unit marked by (fid, sid, pid_{π_i}) to $(\cdot, \cdot, pid_{\pi_i}, pid_{\mathsf{ch}}, \cdot, \cdot)$. It forwards other messages without modification.

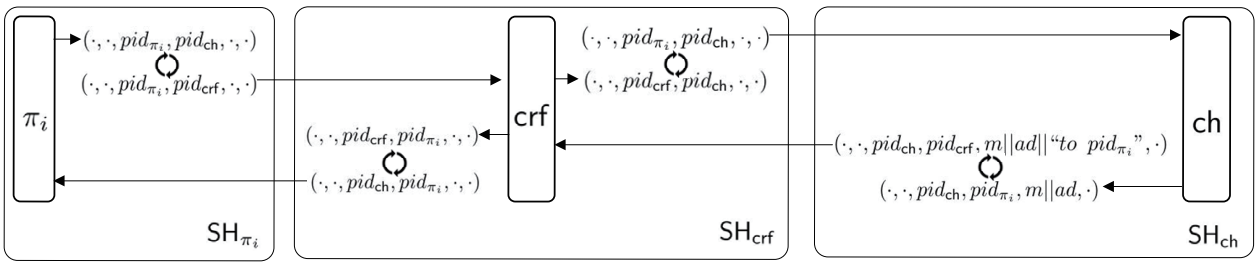

Fig. 3. Operation of the protocol deployed with a CRF. The round arrows represent the transfer operations of shells.

Now we are ready to formalize the operation of cryptographic reverse firewall crf when it composed with protocol $\mathcal{P}$. Following the work [15], we use $\mathcal{P} \circ \mathsf{crf}$ to denote a the protocol $\mathcal{P}$ equipped with a CRF crf. The cryptographic reverse firewall crf runs as follows:

- At the very beginning, the basic unit of crf is invoked.
- When perceiving any abnormality of the inputting messages, crf outputs $(\perp, \perp, pid_{\mathsf{crf}}, \mathcal{E}, \text{"}alarm\text{"}, \text{"}output\text{"})$, and the protocol aborts.
- On receiving $(fid, sid, pid_{\pi_i}, pid_{\mathsf{crf}}, m||ad, \text{"}internal\text{-}output\text{"})$ from a main machine, crf checks if there exists a session-unit marked as (fid, sid, pid_{π_i}). If not, such a new session-unit is generated. Based on the information in the session-unit, crf generates the modified $m'||ad'$ and outputs $(fid, sid, pid_{\mathsf{crf}}, pid_{\mathsf{ch}}, m'||ad', \text{"}internal\text{-}output\text{"})$.
- Similarly, on receiving $(fid, sid, pid_{\mathsf{ch}}, pid_{\mathsf{crf}}, m||ad, \text{"}internal\text{-}output\text{"})$ from a channel machine, crf interprets ad as $ad^*||\text{"}to\ pid_{\pi_j}\text{"}$, and checks if there exists a session-unit marked as (fid, sid, pid_{π_j}). If not, such a new session-unit is generated. Based on the information in the session-unit, crf generates the modified $m'||ad'$ and outputs $(fid, sid, pid_{\mathsf{crf}}, pid_{\mathsf{ch}}, m'||ad', \text{"}internal\text{-}output\text{"})$.
- On receiving messages in type of "*backdoor*", crf handles it as above, and returns messages in type of "*backdoor*".

3.3 Define a CRF by UC-emulation

Based on the above works, we present the new definition of the security of a CRF. Post-Snowden cryptography mainly focuses on the security of the implementations of cryptographic algorithms when they deviate from the corresponding specifications. Such analysis is meaningless if the specifications themselves do not satisfy

the required security. Thus, all works in this paper are based on the premise that the protocol specification is secure, saying, it UC-emulates an ideal functionality $\mathcal{F}$.

The basic idea of the security definition of CRF is that, a CRF can either detect subverted implementations, or it could correct a subverted protocol to UC-emulate the ideal functionality $\mathcal{F}$. We use $\widehat{\mathsf{X}}$ to denote the subverted implementation of X. We first present the game of protocol execution in conventional settings. Readers are referred to Fig. 4 for the game of protocol execution. The machines in dashed boxes may be subverted by adversary $\mathcal{A}$, and the solid lines represent direct communications between ITMs. Concretely,

1. $\mathcal{E}$ is the first Turing machine to be invoked.
2. $\mathcal{E}$ invokes the adversary $\mathcal{A}$.
3. $\mathcal{E}$ generates the inputs of the protocol, and invokes the target protocol $\mathcal{P}$.
4. The target protocol ends when the main machines return their outputs to $\mathcal{E}$.
5. $\mathcal{E}$ outputs its one bit decision $\mathsf{EXEC}_{\mathcal{P},\mathcal{A},\mathcal{E}}$.

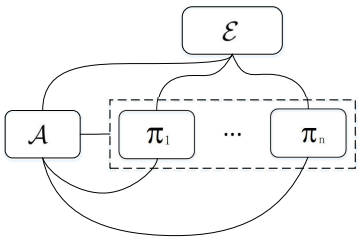

Fig. 4. Execution of protocol in conventional settings.

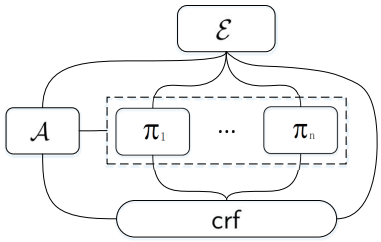

Fig. 5. Execution of protocol equipped with a CRF.

As a special case, when the target protocol is an ideal functionality $\mathcal{F}$, and the adversary is replaced by a simulator $\mathcal{S}$, we denote the output of $\mathcal{E}$ by $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}}$. The security of protocol $\mathcal{P}$ in the conventional setting is defined based on the notion of "UC-emulation" [5]. Formally,

Definition 2. *Let $\mathcal{P}$ be a PPT protocol and $\mathcal{F}$ be an ideal functionality. We say that $\mathcal{P}$* UC-emulates *$\mathcal{F}$, if for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for any PPT environment $\mathcal{E}$, we have*

$$\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{P},\mathcal{A},\mathcal{E}}.$$

That is, for any input, the probability that $\mathcal{E}$ outputs 1 after interacting with $\mathcal{A}$ and $\mathcal{P}$ differs by at most a negligible amount from the probability that $\mathcal{E}$ outputs 1 after interacting with $\mathcal{S}$ and $\mathcal{F}$.

Next, we formulate the game of a protocol in the subversion setting. The main difference here is that the implementations of the target protocol are provided by an adversary, and the protocol is executed with the protection of a CRF. Readers are referred to Fig. 5 for detail. Concretely,

1. $\mathcal{E}$ is the first Turing machine to be invoked.
2. $\mathcal{E}$ invokes the adversary $\mathcal{A}$.
3. $\mathcal{A}$ generates the subverted implementations $\widehat{\mathcal{P}} = \{\widehat{\pi}_1, \cdots, \widehat{\pi}_n, \mathsf{ch}\}$, and submits them to $\mathcal{E}$.
4. $\mathcal{E}$ generates the inputs of the protocol, and invokes the target protocol $\widehat{\mathcal{P}} \circ \mathsf{crf}$.
5. The target protocol ends when crf alarms or the main machines return their outputs to $\mathcal{E}$.
6. $\mathcal{E}$ outputs its one bit decision $\mathsf{EXEC}_{\widehat{\mathcal{P}}\circ\mathsf{crf},\mathcal{A},\mathcal{E}}$.

Before presenting the security definition of a CRF, we define the notion of *robust CRFs*, which refers to that the CRF hardly alarms if all the implementations of protocol obey the specification honestly. Formally,

Definition 3. *A cryptographic reverse firewall* crf *is* robust[3]*, if for any PPT adversary* $\mathcal{A}$ *and PPT environment* $\mathcal{E}$*,* crf *alarms only in a negligible probability when combines with the specification* $\{\pi_1, \cdots, \pi_n\}$ *of protocol* $\mathcal{P}$*.*

Now we are prepared for the definition of the security of a CRF.

Definition 4. *Let* $\mathcal{P}$ *be a PPT protocol which UC-emulates an ideal functionality* $\mathcal{F}$ *in conventional setting. We say a cryptographic reverse firewall* crf guards $\mathcal{P}$ for $\mathcal{F}$*, if (1)* crf *is robust; (2) for all the PPT adversary* $\mathcal{A}$*, the combined protocol* $\widehat{\mathcal{P}} \circ \mathsf{crf}$ *aborts with* crf *alarming in a non-negligible probability, or there exists a PPT simulator* $\mathcal{S}$ *such that for any PPT environment* $\mathcal{E}$*, we have:*

$$\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\widehat{\mathcal{P}}\circ\mathsf{crf},\mathcal{A},\mathcal{E}}.$$

4 Universal Composition of Protocols Deployed with a CRF

As we expect a CRF to serve multiple protocols and multiple sessions, it is necessary to analyze the compositional conditions of protocols with a CRF. Without an explicit formal theorem about composition, even if we prove a CRF is able to guard every protocol session independently, its security in practical application is still unguaranteed. Fortunately, Canetti et al. [6] have proposed the notion of universally composable security with a global setup, which could provide a reference to our work, although their work focused on another topic.

Before presenting our composition theorem, we first introduce the notions of *CRF-subroutine respecting* and *honesty CRF*. These two properties actually achieve that any two sessions have no communication except using a same CRF, which is a little similar to the $\overline{\mathcal{G}}$*-subroutine respecting* proposed by Canetti et al. [6].

Definition 5. *A protocol* $\mathcal{P}$ *is* CRF-subroutine respecting*, if for the instance of* $\mathcal{P}$ *labeled as* $fid_{\mathcal{P}}$ *and* sid_s*, the shell machines* SH_{π_i} *reject all the incoming messages* $(fid, sid, \cdot, \cdot, \cdot, \cdot)$ *that* $\{fid, sid\} \neq \{fid_{\mathcal{P}}, sid_s\}$*.*

Definition 6. *A CRF is called* honest *if it does not change the protocol identity* fid *and session identity* sid *in the transmitted messages.*

Assume the ideal functionality $\mathcal{F}$ is a subroutine of protocol ρ. $\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}$ denotes the protocol which is acquired by replacing the $\mathcal{F}$ in ρ by $\widehat{\mathcal{P}} \circ \mathsf{crf}$. Now we are prepared to present the composition theorem.

Theorem 1. *Let* ρ *and* $\mathcal{P}$ *be PPT protocols, and* $\mathcal{F}$ *be an ideal functionality. If an honest reverse firewall* crf *guards* $\mathcal{P}$ *for* $\mathcal{F}$*, and* $\mathcal{P}$ *is* crf*-subroutine respecting, then for any PPT adversary* $\mathcal{A}$*,* $\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}$ *aborts with* crf *alarming in a non-negligible probability, or there exists a PPT adversary* $\mathcal{S}$ *such that for any PPT environment* $\mathcal{E}$*,*

$$\mathsf{EXEC}_{\rho,\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}},\mathcal{A},\mathcal{E}}.$$

Proof. We need only consider the case when crf does not alarm.

Following the idea in work [5], we take the language of "dummy adversary" to express the CRF's property in a substitutive way. A dummy adversary $\mathcal{D}$ just acts as a "transparent channel" between $\mathcal{E}$ and the machines in protocol. It forwards the backdoor type messages from $\mathcal{E}$ to main machines of protocol $\mathcal{P}$, and forwards the returned message from protocol machines to $\mathcal{E}$. Work [5] provides a rigorous proof that such adjustment from adversary $\mathcal{A}$ to "dummy adversary" $\mathcal{D}$ makes no change to the essence of the protocol security. That is to say, if there exists a PPT environment $\mathcal{E}$ being able to distinguish between an execution of protocol $\widehat{\mathcal{P}} \circ \mathsf{crf}$ with $\mathcal{A}$, and an execution of ideal functionality $\mathcal{F}$ with any PPT simulator $\mathcal{S}$, we can construct another PPT environment $\mathcal{E}'$ being able to distinguish between an execution of protocol $\widehat{\mathcal{P}} \circ \mathsf{crf}$ with dummy adversary $\mathcal{D}$, and an execution of ideal functionality $\mathcal{F}$ with any PPT simulator $\mathcal{S}_{\mathcal{D}}$. Of course, if there is no such $\mathcal{E}$, such $\mathcal{E}'$ does not exist either .

[3] Note that the definition of "robust" in this paper is totally different from the notion of "robust" in work [15].

Following the assumption in Theorem 1, we have that if crf does not alarm, for dummy adversary $\mathcal{D}$ and any PPT environment $\mathcal{E}_{\mathcal{P}}$, there exists a PPT $\mathcal{S}_{\mathcal{D}}$ such that

$$\mathsf{EXEC}_{\mathcal{F},\mathcal{S}_{\mathcal{D}},\mathcal{E}_{\mathcal{P}}} \approx \mathsf{EXEC}_{\widehat{\mathcal{P}}\circ\mathsf{crf},\mathcal{D},\mathcal{E}_{\mathcal{P}}}. \tag{1}$$

Next, we construct a simulator $\mathcal{S}$ out of $\mathcal{S}_{\mathcal{D}}$, such that for any PPT environment $\mathcal{E}$ and adversary $\mathcal{A}$,

$$\mathsf{EXEC}_{\rho,\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}},\mathcal{A},\mathcal{E}}.$$

Readers are referred to Fig. 6 for the operation of $\mathcal{S}$. Concretely, we divide $\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}$ into $\widehat{\mathcal{P}}\circ\mathsf{crf}$ and the rest part of ρ (we use ρ' to denote it). The adversary $\mathcal{A}$ is geared to interact with $\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}$. $\mathcal{S}$ channels the communication between $\mathcal{A}$ and the environment $\mathcal{E}$, and the communication between $\mathcal{A}$ and ρ' without any change. The important operation of $\mathcal{S}$ is that the communication between $\mathcal{A}$ and every instance of $\mathcal{F}$ is "pipelined" with an instance of $\mathcal{S}_{\mathcal{D}}$, saying, messages generated by $\mathcal{A}$ aimed for $\widehat{\mathcal{P}}\circ\mathsf{crf}$ is sent to $\mathcal{S}_{\mathcal{D}}$ as messages from the environment. Incoming messages from $\mathcal{F}$ to $\mathcal{S}_{\mathcal{D}}$ are forwarded without any change. Outputs of $\mathcal{S}_{\mathcal{D}}$ to environment are forwarded to $\mathcal{A}$ as messages from $\widehat{\mathcal{P}}\circ\mathsf{crf}$.

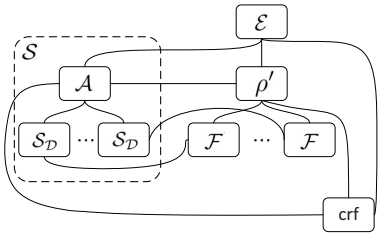

Fig. 6. Construction of simulator $\mathcal{S}$ in the proof of Theorem 1. Full lines denote direct communication between ITMs.

Now we demonstrate that the above construction of $\mathcal{S}$ is available. Based on the "dummy adversary" setting, the adversary $\mathcal{A}$ run by $\mathcal{S}$ behaves exactly like the environment that $\mathcal{S}_{\mathcal{D}}$ expects. We assume that there exists an environment $\mathcal{E}$ which is able to distinguish an execution of $\{\mathcal{S},\rho\}$ and an execution of $\{\mathcal{A},\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}\}$, then we can deduce another environment $\mathcal{E}_{\mathcal{P}}$ which is competent to violate Formula (1). Assume that ρ involves n instances of $\mathcal{F}$, we design $n+1$ hybrids of the game. Concretely,

- **Game** 0: $\mathcal{E}$ aims at distinguishing between a execution of $\{\mathcal{S},\rho\}$ and an execution of $\{\mathcal{A},\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}\}$, where $\mathcal{S}$ is exactly like the case in Fig. 6.
- **Game** $i+1$: is same with **Game** i except that an instance of $\mathcal{F}$ in ρ is replaced by $\widehat{\mathcal{P}}\circ\mathsf{crf}$, the corresponding $\mathcal{S}_{\mathcal{D}}$ connected with such $\mathcal{F}$ in $\mathcal{S}$ is replaced by $\mathcal{D}$.

It is straightforward that $\mathcal{E}$' advantage in **Game** n equals to 0, as the two objects for judging are consistent completely. If there exists an environment $\mathcal{E}$ being able to distinguish an execution of $\{\mathcal{S},\rho\}$ and an execution of $\{\mathcal{A},\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}\}$ with a non-negligible probability, there must exists a pair of game **Game** i and **Game** $i+1$, in which $\mathcal{E}$'s outputs are different in a non-negligible probability.

As shown in Fig. 7, we design $\mathcal{E}_{\mathcal{P}}$ which invokes environment $\mathcal{E}$, adversary $\mathcal{A}$, the caller part of protocol ρ', i executions of $\mathcal{D}$ associated with $\widehat{\mathcal{P}}\circ\mathsf{crf}$, and $(n-i-1)$ executions of $\mathcal{S}_{\mathcal{D}}$ associated with $\mathcal{F}$. The adversary $\mathcal{A}$ and ρ' just operate as normal, except that one pair of their subroutine, i.e., an execution of $\{\mathcal{D},\widehat{\mathcal{P}}\circ\mathsf{crf}\}$, or an execution of $\{\mathcal{S}_{\mathcal{D}},\mathcal{F}\}$, is chosen randomly. If $\mathcal{E}$ succeed in the game to distinguish $\{\mathcal{A},\rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}\}$ and $\{\mathcal{S},\rho\}$ (or its hybrid), $\mathcal{E}_{\mathcal{P}}$ outputs 1; Otherwise $\mathcal{E}_{\mathcal{P}}$ outputs 0. In this case, if $\mathcal{E}_{\mathcal{P}}$ deals with $\{\mathcal{S}_{\mathcal{D}},\mathcal{F}\}$, $\mathcal{E}$ is operates in **Game** i; Else if $\mathcal{E}_{\mathcal{P}}$ deals with $\{\mathcal{D},\widehat{\mathcal{P}}\circ\mathsf{crf}\}$, $\mathcal{E}$ actually operates in **Game** $i+1$. Thus, if $\mathcal{E}$'s outputs are different with a non-negligible probability when in **Game** i and **Game** $i+1$, $\mathcal{E}_{\mathcal{P}}$ is able to distinguish $\{\mathcal{S}_{\mathcal{D}},\mathcal{F}\}$ and $\{\mathcal{D},\widehat{\mathcal{P}}\circ\mathsf{crf}\}$. This is contradiction to Formula (1).

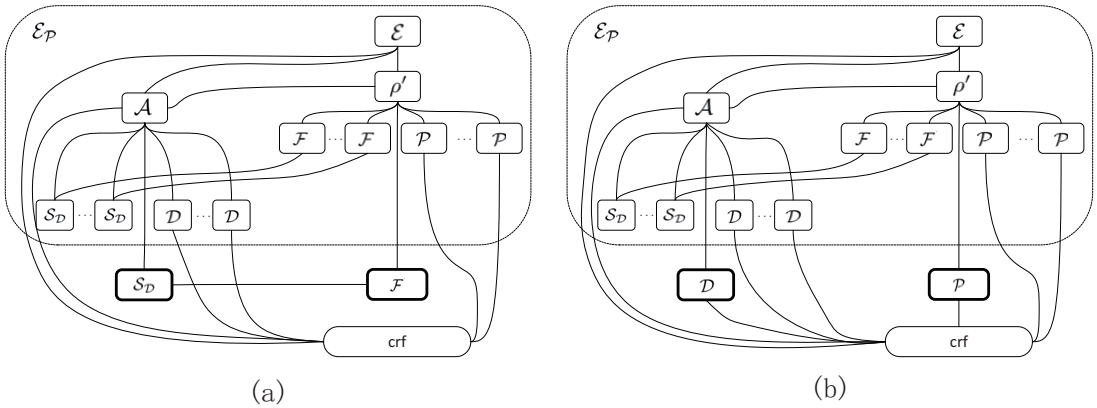

Fig. 7. Construction of $\mathcal{E}_{\mathcal{P}}$ in the proof of Theorem 1. The lines denote direct communication between ITMs. If there exists a PPT $\mathcal{E}$ in the Fig. 6 which is able to distinguish the pair $\{\mathcal{S}, \rho\}$ and $\{\mathcal{A}, \rho^{\mathcal{F}\to\widehat{\mathcal{P}}\circ\mathsf{crf}}\}$, we can construct a $\mathcal{E}_{\mathcal{P}}$ in this figure to violate the security property of crf, i.e., make different output in non-negligible probability when operating in case (a) and case (b).

5 From the New Definition to the Current Definition

In this section we demonstrate that the new definition actually implies the current one. This means that the new definition does not change the basic intuition of CRF, while characterizing the expected properties of a CRF under the idea that "guarding a corrupted system with the minimum trusted module".

For simplicity, here we only consider the PM-adversaries (readers are referred to Section 2.2 for details). The reason for this simplification is that we have allowed a CRF to alarm, in some sense, a CRF can serve as an online watchdog, and implementations provided by adversaries other than PM-adversaries can be easily detected. In this section, we will prove that our definition implies the current definition, that is to say, considering typical subversion attacks, the new definition is at least as strong as the current one. Formally,

Theorem 2. *If a CRF guards protocol $\mathcal{P}$ as defined in Definition 4, then it also satisfies functionality-maintaining, security-preserving, and exfiltration-resistance for PM-adversaries.*

Proof. It is straightforward that for a PM-adversary, the probability that the CRF alarms is negligible. Otherwise we can simply recognize a subverted implementation when the CRF alarms, as it only alarms with a negligible probability when combined with unsubverted implementations as defined.

The functionality and security of a protocol are defined by the indifference between an execution of protocol $\widehat{\mathcal{P}} \circ \mathsf{crf}$ with $\mathcal{A}$, and an execution of ideal functionality $\mathcal{F}$ with $\mathcal{S}$. Now that if a CRF is able to ensure such indifference despite the implementations are corrupted by the adversary, it is straightforward that this CRF satisfies functionality-maintaining and security-preserving.

Assume that there exists a PM-adversary $\mathcal{A}$ being able to break the exfiltration-resistance of a CRF. Then we can build a PPT adversary $\mathcal{B}$ and environment $\mathcal{E}$, for which no PPT simulator $\mathcal{S}$ achieves $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\widehat{\mathcal{P}}\circ\mathsf{crf},\mathcal{B},\mathcal{E}}$, which means the CRF does not guards $\mathcal{P}$. Concretely, when $\mathcal{E}$ operates with $\mathcal{B}$ and $\mathcal{P} \circ \mathsf{crf}$:

1. $\mathcal{E}$ invokes $\mathcal{A}$ to get the subverted implementation $\widehat{\mathcal{P}} = \{\widehat{\pi}_1, \cdots, \widehat{\pi}_n, \mathsf{ch}\}$.
2. $\mathcal{E}$ generates $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, $\mathcal{P}^* \leftarrow \widehat{\mathcal{P}}$; else $\mathcal{P}^* \leftarrow \mathcal{P}$.
3. $\mathcal{E}$ invokes $\mathcal{B}$ and forwards $\mathcal{P}^*$ to $\mathcal{B}$. $\mathcal{B}$ submits them back to $\mathcal{E}$. $\mathcal{E}$ invokes $\mathcal{P}^* \circ \mathsf{crf}$.
4. $\mathcal{B}$ acquires all transcripts of $\widehat{\pi}_i$, and forwards them back to $\mathcal{E}$.
5. $\mathcal{E}$ forwards the transcripts to $\mathcal{A}$, and acquires $\mathcal{A}$'s final output b'.
6. $\mathcal{E}$ returns its final output ($b' = b$).

Thus, if $\mathcal{E}$ interacts with $\{\mathcal{P}^* \circ \mathsf{crf}, \mathcal{B}\}$, according to the definition of exfiltration-resistance (Appendix A) $\Pr[b' = b] = \mathsf{Adv}^{\mathsf{exf}}_{\mathcal{A},\mathsf{crf}}(\lambda) + 1/2$.

Now we consider the case where $\mathcal{E}$ interacts with $\{\mathcal{F}, \mathcal{S}\}$. Since $\mathcal{A}$ is proud-but-malicious, $\widehat{\mathcal{P}}$ and $\mathcal{P}$ are indistinguishable to any PPT simulator $\mathcal{S}$. Thus b has no influence on the returned information from $\mathcal{F}$ and $\mathcal{S}$, i.e., $\Pr[b' = b] = 1/2$. Thus, we have that for any PPT simulator $\mathcal{S}$,

$$\Pr[\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}} \neq \mathsf{EXEC}_{\mathcal{P}^* \circ \mathsf{crf},\mathcal{B},\mathcal{E}}] \geq \mathsf{Adv}^{\mathsf{exf}}_{\mathcal{A},\mathsf{crf}}(\lambda),$$

which completes the proof that the CRF is exfiltration-resistant for PM-adversaries.

6 Construction of a Deterministic CRF

This section presents a detailed construction of CRF. To the best of our knowledge, all of the existing CRFs need to generate a new randomness, to deal with every randomness generated by the underlying protocol. This implies a assumption that a CRF is competent to produce the same amount of randomness to the protocol being protected. Those designs could be regarded as transferring a tough task from users' computers to CRFs.

Our basic framework of CRF construction is more reasonable than existing ones. It takes the idea that although the CRF is still assumed to be unsubverted, it is not expected to generate such a large amount of randomness. Instead, a CRF just needs to get a small amount of fresh and trusted randomness as its keys during the initial phase, and all the following operations are designed to be deterministic. Further more, our construction can even achieve that the CRF's key is allowed to opened to external adversaries, as long as it stays confidential to the implementations of inner protocol parties. In such a situation, the credibility of a CRF is much easier to be realized, as the main source of the risk, probabilistic modules, are shrunken as much as possible. We observe that although an implementation of a probabilistic algorithm is subverted, its output should still have high min-entropy. Otherwise, this subverted implementation will easily be detected in black-box testing. In this case, a CRF can simply alarm when observes a collision, causing the protocol to break off. We take use of this point to achieve our design, which guards a protocol merely using deterministic algorithms, including a module serving as an online watchdog [16].

In this section we only consider the CRF for a class of two-round message-transmission protocol, but under the UC-based definition, the CRF is designed in an extendible setting. Our construction is based on the framework proposed by work [10], which presents a general CRF construction method for message-transmission protocols based on public encryption schemes with special properties.

Preliminary. In order to present a comprehensible introduction of our construction, it is necessary to present several basic notions as a foundation. Considering the page limitation, we put them to appendix. Appendix B.1 introduces a two-round message-transmission protocol based on public-key encryption. Appendix B.2 reviews the notion of rerandomizable encryption and key malleability. Appendix B.3 presents a brief introduction of the CRF construction in work [10].

Our CRF serves for the message-transmission protocol presented in Appendix B.1. We also require that the involved public-key encryption scheme should satisfy (1) *rerandomizable*, with rerandomize function $\mathsf{Rerand} : \mathcal{PK} \times \mathcal{C} \times \{0,1\}^\ell \to \mathcal{C}$; (2) *key malleable*, with rerandomize function $\mathsf{KeyMaul} : \mathcal{PK} \times \{0,1\}^\kappa \to \mathcal{PK}$, and $\mathsf{CKeyMaul} : \mathcal{C} \times \{0,1\}^\kappa \to \mathcal{C}$.

The ideal functionality. In order to analyze the CRF in the new definition, we need to specify the ideal functionality $\mathcal{F}$ corresponding to the message-transmission protocol with an authenticated encryption channel. On receiving the message "*ready*" from Bob, $\mathcal{F}$ informs simulator $\mathcal{S}$, and transfers "*ready*" to Alice after getting $\mathcal{S}$'s agreement. On receiving a message m from Alice, $\mathcal{F}$ informs $\mathcal{S}$, and after receiving a returned agreement, $\mathcal{F}$ forwards m to Bob. Readers are referred to Fig. 8 for details.

Construction. The significant progress of our construction comparing to the design in Appendix B.3 is the acquirement of the coins for rerandomization. We extract r_1 and r_2 from the fixed key of CRF along with the inputs via RO. This transfers the CRF from a probabilistic algorithm to a deterministic one. We present the formal CRF's operation for $\mathcal{P}$ in Fig. 10, and readers are referred to Fig. 9 for a sketch map.

1. Upon receiving "*ready*" from Bob, $\mathcal{F}$ sends "*ready*" to the simulator $\mathcal{S}$.
2. After receiving "*ready-agree*" from $\mathcal{S}$, $\mathcal{F}$ sends "*ready*" to Alice.
3. Upon receiving m from Alice, $\mathcal{F}$ sends "*transmit*" to the simulator $\mathcal{S}$.
4. After receiving "*transmit-agree*" from $\mathcal{S}$, $\mathcal{F}$ sends m to Bob.

Fig. 8. Operation of $\mathcal{F}$ for message-transmission protocol $\mathcal{P}$

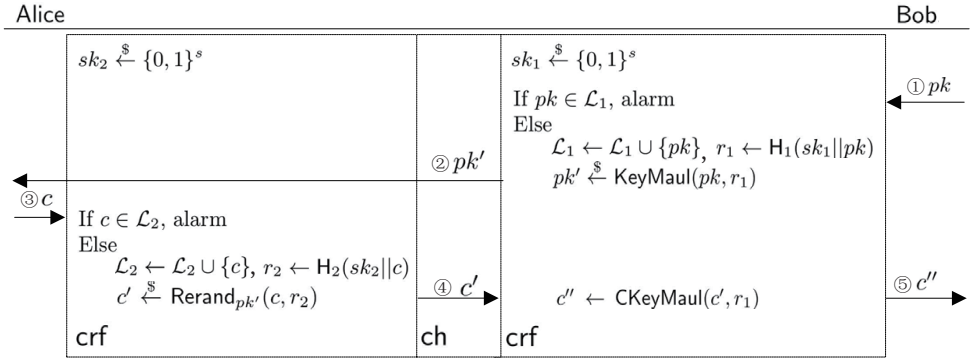

Fig. 9. A sketch map for CRF construction

Remark. In logical sense, we characterize the CRF as a single part as in Section 3.2, but actually in the reality, the CRF is realized by several modules which composed with different protocol parties. It is unpractical and unnecessary to require all the modules to acquire a same key. Thus, in the construction of Fig. 10, the CRF is injected with two keys sk_1 and sk_2, to enable the two parts composed with Bob and Alice, respectively.

We claim that in our construction, the CRF's key sk_1 and sk_2 can be opened to the adversary, as long as it is uniformly random to the subverted parties $\widehat{\mathsf{Alice}}$ and $\widehat{\mathsf{Bob}}$. This property provides us with a possible operation pattern of CRF, i.e., injecting public randomness into the CRF after the deployment of implementations of protocol parties.

Theorem 3. *The CRF in Fig. 10 guards $\mathcal{P}$ for $\mathcal{F}$, if H_1 and H_2 are modeled as random oracles.*

Proof. We first demonstrate that crf is robust. Given that PE is a probabilistic encryption scheme, for $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{KeyGen}(\lambda)$ and $m_0, m_1 \xleftarrow{\$} \mathcal{M}$, where $\mathcal{M}$ is the plaintext space, we have $\Pr[pk_0 = pk_1] \leq \mathsf{negl}(\lambda)$, and $\Pr[\mathsf{Enc}_{pk_a}(m_b) = \mathsf{Enc}_{pk_c}(m_d)] \leq \mathsf{negl}(\lambda)$ $(a, b, c, d \in \{0, 1\}$ and $(a, b) \neq (c, d))$. As crf only alarms when the inputting messages is not consistent to the standard form, or there is collision among the input public-keys or ciphertexts, when combining with the specification, it alarms in a negligible probability.

Thus, we only need to consider the case where CRF does not alarm. Following the proof of Theorem 1, here we still take the method of "dummy adversary", saying, we will design a PPT simulator $\mathcal{S}$ such that any PPT environment $\mathcal{E}$ is unable to distinguish an execution of protocol $\widehat{\mathcal{P}} \circ \mathsf{crf}$ with dummy adversary $\mathcal{D}$, and an execution of ideal functionality $\mathcal{F}$ with $\mathcal{S}$. Work [5] has presented a complete proof that such adjustment from adversary to dummy adversary actually makes no influence to the security property of protocols.

We construct a PPT simulator $\mathcal{S}$ which takes the following strategy in a session.

1. On receiving "*ready*" from $\mathcal{F}$, $\mathcal{S}$ generates $(pk^*, sk^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$, passes pk^* to $\mathcal{E}$, and sends "*ready-agree*" to $\mathcal{F}$.
2. On receiving "*transmit*" from $\mathcal{F}$, $\mathcal{S}$ generates $\mathsf{Rerand}_{pk^*}(\mathsf{Enc}(0), U_\ell)$, passes it to $\mathcal{E}$, and sends "*transmit-agree*" to $\mathcal{F}$.

We present the following hybrids of the CRF:

Cryptographic reverse firewall for protocol $\mathcal{P}$

<u>Initial phase(sk_1, sk_2)</u> $\quad\quad$ $\backslash\backslash sk_1 \xleftarrow{\$} \{0,1\}^s$, $sk_2 \xleftarrow{\$} \{0,1\}^s$

Establish two empty list $\mathcal{L}_1$ and $\mathcal{L}_2$

<u>Unavaliable message$(\cdot)$</u>

When receiving messages not consistent to the forms of outputs from the specification, alarm

<u>Key malleation $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Bob}}, pid_{\mathsf{crf}}, pk||$"*public-key to* Alice", "*internal-output*"$)$</u>

Generate a session-unit marked as $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Bob}})$
If $pk \in \mathcal{L}_1$, alarm
Else
  $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \{pk\}$, $r_1 \leftarrow \mathsf{H}_1(sk_1||pk)$, store r_1 in the session-unit.
  $pk' \leftarrow \mathsf{KeyMaul}(pk, r_1)$
  Output $(fid_{\mathcal{P}},\ sid_n, pid_{\mathsf{crf}}, pid_{\mathsf{ch}}, pk'||$"*public-key to* Alice", "*internal-output*"$)$

<u>Receive key $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{ch}}, pid_{\mathsf{crf}}, pk'||$"*public-key to* Alice", "*internal-output*"$)$</u>

Generate a session-unit marked as $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Alice}})$
Record pk' in the session-unit
Output $(fid_{\mathcal{P}},\ sid_n, pid_{\mathsf{crf}}, pid_{\mathsf{Alice}}, pk'||$"*public-key to* Alice", "*internal-output*"$)$

<u>Ciphertext rerandomization $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Alice}}, pid_{\mathsf{crf}}, c||$"*cipertext to* Bob", "*internal-output*"$)$</u>

If there is no session-unit marked as $\{fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Alice}}\}$, return $\perp$
Else
  Invoke the public-key pk' from the session-unit marked as $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Alice}})$
  If $c \in \mathcal{L}_2$, alarm
  Else
    $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup \{c\}$, $r_2 \leftarrow \mathsf{H}_2(sk_2||c)$
    $c' \leftarrow \mathsf{Rerand}_{pk'}(c, r_2)$
    Output $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{crf}}, pid_{\mathsf{ch}}, c'||$"*cipertext to* Bob", "*internal-output*"$)$

<u>Inverse key malleation $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{ch}}, pid_{\mathsf{crf}}, c'||$"*ciphertext to* Bob", "*internal-output*"$)$</u>

If there is no session-unit marked as $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Bob}})$, returns $\perp$
Else
  Invoke r_1 from the session-unit marked as $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{Bob}})$
  $c'' \leftarrow \mathsf{CKeyMaul}(c', r_1)$
  Output $(fid_{\mathcal{P}}, sid_n, pid_{\mathsf{crf}}, pid_{\mathsf{Bob}}, c''||$"*cipertext to* Bob", "*internal-output*"$)$

Fig. 10. Operation of crf for protocol $\mathcal{P}$

- **Hybrid 0**: It is identical to Fig. 10.
- **Hybrid 1**: It runs in the same way as **Hybrid 0**, except that r_1 in Key malleation and Inverse key malleation are substituted by $r_1' \xleftarrow{\$} \{0,1\}^\kappa$.
- **Hybrid 2**: It runs in the same way as **Hybrid 1**, except that r_2 in Ciphertext rerandomization is substituted by $r_2' \xleftarrow{\$} \{0,1\}^\ell$.
- **Hybrid 3**: It runs in the same way as **Hybrid 2**, except that in Key malleation, pk' is replaced by pk^*, and $(pk^*, sk^*) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- **Hybrid 4**: It runs in the same way as **Hybrid 3**, except that in Ciphertext rerandomization, c' is replaced by $c^* \leftarrow \mathsf{Rerand}_{pk'}(\mathsf{Enc}_{pk'}(0), U_\ell)$.

We denote the environment $\mathcal{E}$'s output after interacting with $\mathcal{D}$ and protocol equipped **Hybrid i** as $\mathsf{EXEC}_{\mathcal{P}\circ\mathsf{Hyb}i,\mathcal{D},\mathcal{E}}$, and the corresponding advantage as

$$\mathsf{Adv}^i_{\mathcal{E}}(\lambda) = \frac{1}{2}\Big|\Pr[\mathsf{EXEC}_{\mathcal{F},\mathcal{S}_\mathcal{P},\mathcal{E}} = 1] - \Pr[\mathsf{EXEC}_{\mathcal{P}\circ\mathsf{Hyb}i,\mathcal{D},\mathcal{E}} = 1]\Big|.$$

Lemma 1. *For any PPT environment $\mathcal{E}$, we have*

$$\Big|\mathsf{Adv}^1_{\mathcal{E}}(\lambda) - \mathsf{Adv}^0_{\mathcal{E}}(\lambda)\Big| \le \frac{n_1}{2^s} + \frac{n_2}{2^\delta},$$

where n_1 and n_2 are numbers of queries from the subverted implementation and the environment/dummy adversary. δ is the min-entropy of public-key pk produced by the potentially subverted party $\widehat{\mathsf{Bob}}$.

Proof (of Lemma 1). Based on the definition of RO, **Hybrid 1** only differs from **Hybrid 0** when

- Event 1: subverted implementation submits $sk_1||pk$ to RO;
- Event 2: environment $\mathcal{E}$ or dummy adversary $\mathcal{D}$ submits $sk_1||pk$ to RO.

As sk_1 is generated at the initial phase of CRF, thus, it stays uniformly random to the implementations of protocol parties. So we have

$$\Pr[\text{Event 1}] \le \frac{n_1}{2^s}.$$

Now we explore the probability of Event 2. We claim that although the protocol party Bob is subverted, its output pk should still reach a high min-entropy, otherwise, crf will alarm in a non-negligible probability. Concretely, if we assume that

$$\max_{pk^*} \Pr[pk = pk^*, (pk, sk_1) \leftarrow \widehat{\mathsf{Keygen}}(\lambda)] = 2^{-\delta} = \frac{1}{\mathsf{ploy}(\lambda)},$$

then, within q rounds of the protocol, the probability of crf alarms:

$$\Pr[\mathsf{crf}\text{ alarms}] \ge \frac{1}{2}q(q-1)\cdot 2^{-2\delta} = \frac{q(q-1)}{2\cdot \mathsf{ploy}^2(\lambda)}$$

Thus, if crf only alarms in a negligible probability, we have $\delta \le \mathsf{negl}(\lambda)$. In this case,

$$\Pr[\text{Event 2}] \le n_2 2^{-\delta} < \mathsf{negl}(\lambda).$$

So we have

$$\Big|\mathsf{Adv}^1_{\mathcal{E}}(\lambda) - \mathsf{Adv}^0_{\mathcal{E}}(\lambda)\Big| \le \Pr[\text{Event 1}] + \Pr[\text{Event 2}] < \frac{n_1}{2^s} + \frac{n_2}{2^\delta}. \tag{2}$$

Based on the same idea, we present Lemma 2 without detailed proof.

Lemma 2. *For any PPT environment $\mathcal{E}$, we have*

$$\left|\mathsf{Adv}^2_{\mathcal{E}}(\lambda) - \mathsf{Adv}^1_{\mathcal{E}}(\lambda)\right| \leq \frac{n_3}{2^s} + \frac{n_4}{2^\sigma},$$

where n_3 and n_4 are numbers of queries from the subverted implementation and the environment/dummy adversary, respectively; σ is the min-entropy of ciphertext c produced by the potentially subverted party $\widehat{\mathsf{Alice}}$.

Based on the property of key malleable encryption in Definition 11, when the coin used in $\mathsf{KeyMaul}$ is a uniformly randomness, the output key from $\mathsf{KeyMaul}$ is uniformly random distributed over the public-key space. Thus, we could present Lemma 3 directly,

Lemma 3. *For any PPT environment $\mathcal{E}$, we have $\mathsf{Adv}^3_{\mathcal{E}}(\lambda) = \mathsf{Adv}^2_{\mathcal{E}}(\lambda)$.*

Based on the property of rerandomizable encryption in Definition 10, when the coin used in Rerand is a uniformly randomness, $(c; \mathsf{Rerand}_{pk}(c, U_\kappa))$ is computationally indistinguishable from $(c; \mathsf{Rerand}_{pk}(\mathsf{Enc}_{pk}(0), U_\kappa))$, Thus. we could present Lemma 4 directly,

Lemma 4. *For any PPT environment $\mathcal{E}$, $\left|\mathsf{Adv}^4_{\mathcal{E}}(\lambda) - \mathsf{Adv}^3_{\mathcal{E}}(\lambda)\right| \leq \mathsf{negl}(\lambda)$.*

It is obviously that $\mathsf{Adv}^4_{\mathcal{E}}(\lambda) = 0$. Summing up Lemma 1-4, we reach the conclusion

$$\mathsf{EXEC}_{\mathcal{F},\mathcal{S}_{\mathcal{P}},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{P}\circ\mathsf{crf},\mathcal{D},\mathcal{E}},$$

which means that $(\mathcal{S}, \mathcal{F})$ makes an ideal emulation to $(\mathcal{D}, \widehat{\mathcal{P}} \circ \mathsf{crf})$.

Acknowledgment. The research in this paper is mainly done when the first author (Geng Li) was a Ph. D candidate in School of Cyber Science and Technology, Beihang University. This work is partly supported by the National Natural Science Foundation of China (61972017) and the Fundamental Research Funds for the Central Universities (YWF-21-BJ-J-1040).

References

1. Ateniese, G., Francati, D., Magri, B., Venturi, D.: Public immunization against complete subversion without random oracles. In: ACNS. vol. 11464, pp. 465–485 (2019)
2. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: ACM CCS. pp. 364–375 (2015)
3. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In: ACM CCS. pp. 1431–1440 (2015)
4. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: CRYPTO. vol. 8616, pp. 1–19 (2014)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145 (2001)
6. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: TCC. vol. 4392, pp. 61–85 (2007)
7. Canetti, R., Rabin, T.: Universal composition with joint state. In: CRYPTO. vol. 2729, pp. 265–281 (2003)
8. Chen, R., Mu, Y., Yang, G., Susilo, W., Guo, F., Zhang, M.: Cryptographic reverse firewall via malleable smooth projective hash functions. In: ASIACRYPT. vol. 10031, pp. 844–876 (2016)
9. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: FSE. vol. 9054, pp. 579–598 (2015)
10. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls - secure communication on corrupted machines. In: CRYPTO. vol. 9814, pp. 341–372 (2016)
11. Fischlin, M., Mazaheri, S.: Self-guarding cryptographic protocols against algorithm substitution attacks. In: CSF. pp. 76–90 (2018)
12. Hofheinz, D., Shoup, V.: GNUC: A new universal composability framework. J. Cryptol. **28**(3), 423–508 (2015)
13. Li, G., Liu, J., Zhang, Z.: Security against subversion in a multi-surveillant setting. In: ACISP. vol. 11547, pp. 419–437 (2019)

14. Li, G., Liu, J., Zhang, Z.: A more realistic analysis of mass surveillance - security in multi-surveillant settings. IET Information Security. **2020**, 643–651 (2020)
15. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: EUROCRYPT. vol. 9057, pp. 657–686 (2015)
16. Russell, A., Tang, Q., Yung, M., Zhou, H.: Cliptography: Clipping the power of kleptographic attacks. In: ASIACRYPT. vol. 10032, pp. 34–64 (2016)
17. Russell, A., Tang, Q., Yung, M., Zhou, H.: Generic semantic security against a kleptographic adversary. In: ACM CCS. pp. 907–922 (2017)
18. Young, A.L., Yung, M.: The dark side of "black-box" cryptography or: Should we trust capstone? In: CRYPTO. vol. 1109, pp. 89–103 (1996)
19. Young, A.L., Yung, M.: Kleptography: Using cryptography against cryptography. In: EUROCRYPT. vol. 1233, pp. 62–74 (1997)
20. Young, A.L., Yung, M.: The prevalence of kleptographic attacks on discrete-log based cryptosystems. In: CRYPTO. vol. 1294, pp. 264–276 (1997)
21. Young, A.L., Yung, M.: Malicious cryptography: Kleptographic aspects. In: CT-RSA. vol. 3376, pp. 7–18 (2005)

A The Current Definition of a CRF

Mironov and Stephens-Davidowitz [15] characterized the expect properties of a CRF by functionality-maintaining, security-preserving and exfiltration-resistance.

Definition 7. *(Functionality-maintaining) For any CRF* crf *and any party* A*, let* $\mathsf{A} \circ \mathsf{crf}^1 = \mathsf{A} \circ \mathsf{crf}$*, and for* $k \geq 2$*,* $\mathsf{A} \circ \mathsf{crf}^k = (\mathsf{A} \circ \mathsf{crf}^{k-1}) \circ \mathsf{crf}$*. For a protocol* $\mathcal{P}$ *that satisfies functionality requirement* $\mathcal{FR}$*, for any polynomial bounded* $k > 1$*, if* $\mathsf{A} \circ \mathsf{crf}^k$ *maintains* $\mathcal{FR}$ *for* A *in* $\mathcal{P}$*, we say that* crf *maintains* $\mathcal{FR}$ *for* A *in* $\mathcal{P}$*.*

$\mathcal{P}_{\mathsf{A} \to \overline{\mathsf{A}} \circ \mathsf{crf}}$ denotes the protocol in which party A is replaced by the combination of crf and a subverted party $\overline{\mathsf{A}}$. Besides, we denote a functionality-maintaining party by $\widehat{\mathsf{A}}$ and an arbitrary subverted party by $\overline{\mathsf{A}}$.

Definition 8. *(Security-preserving) For a protocol* $\mathcal{P}$ *that satisfies security requirements* $\mathcal{SR}$*, and a CRF* crf*, we say that*

- crf strongly preserves $\mathcal{SR}$ *for* A *in* $\mathcal{P}$ *if the protocol* $\mathcal{P}_{\mathsf{A} \to \overline{\mathsf{A}} \circ \mathsf{crf}}$ *satisfies* $\mathcal{SR}$*;*
- crf weakly preserves $\mathcal{SR}$ *for* A *in* $\mathcal{P}$ *if the protocol* $\mathcal{P}_{\mathsf{A} \to \widehat{\mathsf{A}} \circ \mathsf{crf}}$ *satisfies* $\mathcal{SR}$*.*

The exfiltration-resistance is defined based on the leakage game which is presented in Fig. 11. We assume the protocol has two parties A and B. The advantage of an adversary $\mathcal{A}$ in **Game** LEAK is defined as

$$\mathsf{Adv}^{\mathsf{LEAK}}_{\mathcal{A},\mathsf{crf}}(\lambda) \stackrel{def}{=} \left| \Pr\left[\mathsf{LEAK}(\mathcal{P}, \mathsf{A}, \mathsf{B}, \mathsf{crf}, \lambda) = 1\right] - \frac{1}{2} \right|.$$

Game LEAK $(\mathcal{P}, \mathsf{A}, \mathsf{B}, \mathsf{crf}, \lambda)$
$(st_{\overline{\mathsf{A}}}, \overline{\mathsf{A}}, \overline{\mathsf{B}}, I) \leftarrow \mathcal{A}(1^\lambda)$
$b \stackrel{\$}{\leftarrow} \{0, 1\}$
IF $b = 1$, $\mathsf{A}^* \leftarrow \overline{\mathsf{A}} \circ \mathsf{crf}$
ELSE, $\mathsf{A}^* \leftarrow \mathsf{A} \circ \mathsf{crf}$
$\Gamma^* \leftarrow \mathcal{P}_{\mathsf{A} \to \mathsf{A}^*, \mathsf{B} \to \overline{\mathsf{B}}}(I)$
$b^* \leftarrow \mathcal{A}(\Gamma^*, st_{\overline{\mathsf{B}}})$
RETURN $(b = b^*)$

Fig. 11. The leakage game LEAK. $st_{\overline{\mathsf{A}}}$ and $st_{\overline{\mathsf{B}}}$ are the state of $\overline{\mathsf{A}}$ and $\overline{\mathsf{B}}$. I is a valid input for $\mathcal{P}$, and Γ^* is the transcript produced by protocol $\mathcal{P}_{\mathsf{A} \to \mathsf{A}^*, \mathsf{B} \to \overline{\mathsf{B}}}(I)$.

Definition 9. *(Exfiltration-resistance) For a protocol that satisfies functionality requirements $\mathcal{FR}$ and a reverse firewall* crf*, we say that*

- crf *is* strongly exfiltration-resistant *for party* A *against party* B *in the protocol* $\mathcal{P}$ *if for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{LEAK}}_{\mathcal{A},\mathsf{crf}}(\lambda)$ *is negligible in the security parameter* λ*; and*
- crf *is* weakly exfiltration-resistant *for party* A *against party* B *in the protocol* $\mathcal{P}$ *if for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{LEAK}}_{\mathcal{A},\mathsf{crf}}(\lambda)$ *is negligible in the security parameter* λ *provided that* $\overline{\mathsf{A}}$ *maintains* $\mathcal{FR}$ *for party* A*.*

In the special case where B *is empty, we say that* crf *is* exfiltration-resistant against eavesdroppers*.*

B CRF Construction in Work [10]

B.1 Two-round Message-Transmission Protocol

Let $\mathsf{PE} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. $\mathsf{Keygen} : 1^\lambda \to (\mathcal{PK}, \mathcal{SK})$ is key generation algorithm, where $\mathcal{PK}$ and $\mathcal{SK}$ are public-key space and secret-key space, respectively. $\mathsf{Enc} : \mathcal{PK} \times \mathcal{M} \to \mathcal{C}$ is encryption algorithm, where $\mathcal{M}$ and $\mathcal{C}$ are plaintext space and ciphertext space, respectively. $\mathsf{Dec} : \mathcal{SK} \times \mathcal{C} \to \mathcal{M}$ is decryption algorithm. We define the correctness of PE as: for any $m \in \mathcal{M}$ and $(pk, sk) \leftarrow \mathsf{Keygen}(1^\lambda)$, we have $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$. And we define the CPA-security of PE as for any adversarially chosen pair of plaintexts (m_0, m_1), $(pk, \mathsf{Enc}_{pk}(m_0))$ and $(pk, \mathsf{Enc}_{pk}(m_1))$ are computational indistinguishable.

The basic form of a two-round public-key based message-transmission protocol is like this: Bob generates $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends pk to Alice. Alice encrypts massage m by $c \leftarrow \mathsf{Enc}_{pk}(m)$. After getting c, Bob decrypts it via $m \leftarrow \mathsf{Dec}_{sk}(c)$.

B.2 Rerandomizable Encryption and Key Malleability Encryption

Rerandomizable encryption refers to the public-key encryption whose ciphertexts can be rerandomized to a new one, without bringing influence to the normal decryption. Formally,

Definition 10. *[10] A public-key encryption scheme is* rerandomizable *if there is a PPT algorithm* $\mathsf{Rerand} : \mathcal{PK} \times \mathcal{C} \times \{0,1\}^\kappa \to \mathcal{C}$*, for any ciphertext* c *such that* $\mathsf{Dec}_{sk}(c) \neq \perp$*, we have: (1)* $\mathsf{Dec}_{sk}(\mathsf{Rerand}_{pk}(c, U_\kappa)) = \mathsf{Dec}_{sk}(c)$*, (2)* $(c; \mathsf{Rerand}_{pk}(c, U_\kappa))$ *is computationally indistinguishable from* $(c; \mathsf{Rerand}_{pk}(\mathsf{Enc}_{pk}(0), U_\kappa))$*.*

Key malleable encryption refers to encryption whose public-key can be transfered to a new one, and there exists an efficient algorithm mapping ciphertexts under the new key-pair to ciphertexts under initial key-pair. Formally,

Definition 11. *[10] A public-key encryption scheme is key malleable if: (1) the output of* KeyGen *is distributed uniformly over the space of valid keys; (2) for each public-key* pk *there is a unique associated private key* sk*; and (3) there is a pair of efficient algorithms* $\mathsf{KeyMaul}$ *and* $\mathsf{CKeyMaul}$ *that behave as follows.* $\mathsf{KeyMaul}$ *takes as inputs a public-key* pk *and a randomness* $r \xleftarrow{\$} \{0,1\}^\ell$ *and returns a new public-key* pk' *whose distribution is uniformly random over the public-key space. Let* (sk, pk) *be a private-key/public-key pair, and let* (sk', pk') *be the unique pair associated with randomness* r *such that* $pk' = \mathsf{KeyMaul}(pk, r)$*. Then,* $\mathsf{CKeyMaul}$ *takes as inputs a ciphertext* c *and randomness* r *and returns* c' *such that* $\mathsf{Dec}_{sk'}(c) = \mathsf{Dec}_{sk}(c')$*.*

For example, Elgamal encryption is both rerandomizable and key malleable. Such encryption can also be achieved via universal hash proof function [10].

B.3 CRF Construction for Two-round Message-Transmission Protocols

The work [10] presents a CRF construction, which takes the idea that rerandomizing the public-key and the ciphertext by KeyMaul and Rerand, respectively. After receiving a processed ciphertext, the CRF adjusts it to match the original secret key. Readers are referred to Fig. 12 for detailed formalization. Concretely,

1. When receiving pk from Bob, if pk is not in the standard form, CRF alarms; else, it generates $r_1 \xleftarrow{\$} \{0,1\}^\ell$ and $pk' \xleftarrow{\$} \mathsf{KeyMaul}(pk, r_1)$, and outputs pk' to channel.

2. When receiving c from Alice, if c is not in the standard form, CRF alarms; else, it generates $r_2 \xleftarrow{\$} \{0,1\}^\kappa$ and $c' \xleftarrow{\$} \mathsf{Rerand}_{pk'}(c, r_2)$, and outputs c' to channel.

3. When receiving c' from channel, if c' is not in the standard form, CRF alarms; else, it invokes r_1, generates $c'' \leftarrow \mathsf{CKeyMaul}(c', r_1)$, and outputs c'' to Bob.

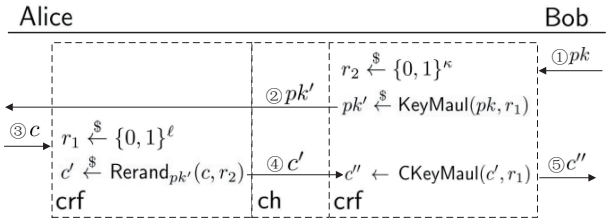

Fig. 12. CRF based on rerandomizable encryption