Universally Composable Subversion-Resilient Authenticated Key Exchange

Jiahao Liu[1], Yi Wang[1], Rongmao Chen[1(✉)], Xinyi Huang[2], Jinshu Su[3], and Moti Yung[4]

[1] College of Computer Science and Technology, National University of Defense Technology, Changsha, China
{liujiahao14,wangyi14,chromao}@nudt.edu.cn
[2] Jinan University, Guangzhou, China
xyhuang81@gmail.com
[3] Academy of Military Science, Beijing, China
sjs@nudt.edu.cn
[4] Google LLC & Columbia University, New York, USA
motiyung@gmail.com

Abstract. Subversion-resilient cryptography has garnered increasing attention in recent years due to growing concerns about cryptographic subversions in real-world applications. Among the existing countermeasures, the notion of cryptographic reverse firewalls (RFs), initially proposed by Mironov and Stephens-Davidowitz (EUROCRYPT 2015) and later extended by Chakraborty et al. (EUROCRYPT 2022) to the universally composable (UC) model, has proven to be a powerful tool for building subversion-resilient cryptographic protocols. In this work, we focus on designing subversion-resilient authenticated key exchange (AKE) protocols, which are critical components of secure Internet communication. We present the first generic framework for subversion-resilient UC-secure AKE protocols leveraging RFs. Inspired by the state-of-the-art advancements by Chakraborty et al. (ASIACRYPT 2024), we address subversions: where a party's implementation is covertly altered to exfiltrate secrets or behave unpredictably when triggered by adversarial inputs. A key contribution of our work is the introduction of a new AKE functionality which, for the first time, incorporates security against key control, an essential aspect of achieving subversion resilience. We also provide a concrete instantiation of our framework, demonstrating its feasibility in practice. Notably, the RFs in our proposed AKE protocol are *transparent*, an important property of RF as defined originally, which allows deployment of RF without all parties explicitly knowing about it and allows robust security. Achieving transparency for RFs has been widely regarded as challenging, particularly when addressing broader subversion attacks (e.g., input-trigger attacks) in the UC model. Our approach, thus, not only advances the state of AKE protocol design, but also offers insights into building other subversion-resilient protocols in the UC model using transparent RFs.

G. Hanaoka and B.-Y. Yang (Eds.): ASIACRYPT 2025, LNCS 16246, pp. 103–136, 2026.
https://doi.org/10.1007/978-981-95-5096-8_4

Keywords: Authenticated Key Exchange · Subversion Resilience · Transparent Reverse Firewalls · Universally Composable Security

1 Introduction

Authenticated Key Exchange (AKE) protocols enable two parties to establish a session key securely over an insecure communication channel and mutually authenticate the parties' identities. Hence these protocols are essential for secure communications in various applications, including secure messaging, online transactions, and establishing secure connections in networks (i.e., TLS/SSL).

Traditional security models for AKE protocols implicitly assume that protocol implementations strictly adhere to the specifications, and do not consider such a powerful adversary that subverts the honest implementation to embed cryptographic trapdoors so that it can retrieve sensitive information covertly, thus undermining the security of the AKE protocols. These cryptographic subversion attacks, identified as kleptographic attacks by Young and Yung [56,57] in the 1990s, initially regarded somewhat theoretical, until Snowden's revelations in 2013 exposed the potential usage of such attack in mass surveillance by intelligence agencies [48], which rekindled the attention of the cryptographic community [1,4,7,8,12,24,29,34,43,55]. In another investigation, real-world AKE protocols, such as the TLS handshake, have been shown to be vulnerable to subversion attacks [13].

Subversion-Resilient UC Model. Note that an AKE protocol is often integrated as a sub-module of complex protocols such as TLS [31], Signal [16,27] and WireGuard [40], where multiple instances run concurrently, rather than being used in a standalone fashion. It is, therefore, highly desirable to construct AKE protocols that offer composable security when considering the presence of subversion attacks.

The study of universally composable (UC) subversion-resistant cryptography was initiated by Chakraborty et al. [23]. They extended the notion of cryptographic reverse firewall (RF) proposed by Mironov and Stephens-Davidowitz [47], which has been widely adopted to defend against subversion attacks for various primitives or protocols [4,14,18–20,22,25,30], to the setting of the UC model (hereafter referred to as the srUC model). In this model, each party consists of a core and an RF. The core holds the secret inputs of the party and generates the messages of the protocol, while the RF does not have access to the secrets of the core and sanitizes the incoming/outgoing messages for the core with randomness.

At ASIACRYPT 2024, Chakraborty et al. [22] extended the srUC model to the unauthenticated setting, and constructed the first password-authenticated key exchange (PAKE) protocol under this model. The core idea is to first realize a subversion-resilient PAKE protocol assuming the existence of an ideal functionality $\mathcal{F}_{\mathsf{SAT}}$ for sanitizable authenticated communication, and then to transform it into a new protocol over unauthenticated channels via "split functionalities" [5].

In particular, they proposed a protocol that realizes the split version of $\mathcal{F}_{\mathsf{SAT}}$ in the unauthenticated setting and is an essential component of the transformation. In the message authentication phase of this protocol, the RF must ask the core to generate a signature for the sanitized message. That is, the RF in this PAKE protocol is not transparent, and the core is aware of the existence of its RF.

Transparency of RFs. It is worth mentioning that in the original proposal of RFs [47], Mironov and Stephens-Davidowitz stressed that they were interested in RFs that preserve the functionality of an already working protocol, and the proposed RFs are **all transparent** in the sense that the behavior of a party composed of an honest core and an RF is identical to the behavior of that honest core. Since a transparent RF does not change the behavior of the core, it could be deployed as a security enhancement module on the fly and without drastic changes throughout the overall system. Thus, it is highly preferable to design subversion-resilient protocols with transparent RFs in practice if we hope to actually employ the mechanism.

The above observation aligns with a recent work by Arnold et al. [2], which argues that non-transparent RFs, if corrupted, would drastically weaken the security guarantee provided by the srUC model. To address this, they proposed a simpler model that captures RFs within the *plain* UC model, enabling the reuse of existing tools (e.g., the composition theorem) and UC-secure primitives. They proved that to establish the UC security of a protocol against subversion attacks under their model, it suffices to demonstrate that the underlying RFs satisfy certain properties, including transparency. However, the subversion attacks considered in their model are restricted to subverting the randomness used by the cores. Other types of subversion attacks, particularly input-trigger attacks—where a subverted core behaves in an unexpected way upon receiving a triggering message crafted by the adversary—fall outside the scope of Arnold et al.'s model. Note that these attacks have been thoroughly examined in the design of subversion-resilient PAKE protocols in the srUC model [22].

Motivated by the aforementioned observations and driven by the goal of designing subversion-resilient AKE protocols, we pose the following question, left open by prior works, as the central focus of this work:

> *Is it possible to design AKE protocols with transparent RFs in the* srUC *model?*

It is worth noting that AKE protocols, like PAKE protocols, also run over unauthenticated channels. Hence, a key challenge in addressing the aforementioned question lies in mitigating the subversion attacks captured by the srUC model in the unauthenticated setting, as studied by Chakraborty et al. [22], while leveraging transparent RFs—a goal that was not achieved in prior work.

1.1 Our Contributions

This paper provides a positive answer to the above question by constructing the first subversion-resilient AKE protocol srAKE with transparent RFs under

Table 1. Security properties captured by ideal functionalities for AKE.

	[17,39]	[41]	[37,52]	**Ours**
Forward Secrecy	full	full	weak	weak
KCI Security	×	✓	✓	✓
Security Against Key Control	×	×	×	✓

the srUC model [23] (in the setting of unauthenticated links). In particular, we show a generic framework built on key encapsulation mechanisms (KEMs) and commitment schemes. The starting point of our construction is a highly efficient key exchange protocol by Cohn-Gordon et al. [28]. In the next section, we give a high-level explanation on how this protocol can be modified to meet the requirements of **transparent RFs** and to **defend against input-trigger attacks**. To demonstrate the feasibility of the proposed framework, we provide a concrete instantiation of the underlying KEMs and commitment schemes.

We also present a variant of the AKE functionality by Jarecki et al. [41] that, for the first time, captures *security against key control*. This property requires that no party (even a corrupted or subverted one) can have any control over the session key. Recall that the goal of subversion-resilience is to prevent the leakage of sensitive information and to maintain the security in the presence of subversion attacks. It seems that security against key control does not contribute to subversion-resilience. However, when security against key control is not guaranteed, it is possible for a subverted party to control, say, certain bits of the session key as requested by the subversion attacker, and the attacker would obtain the key information. Thus, Dodis et al. [30] emphasized that security against key control is essential for a key-agreement protocol to be subversion-resilient.

2 Results Overview and Related Works

2.1 AKE Functionality

Before designing UC-secure AKE protocols, it is necessary to determine the target ideal functionality, as there are a number of ideal functionalities for AKE [17,37,39,41,52] to date. Table 1 shows the comparison of these functionalities in terms of security properties. One can note that none of them imply security against key control. Our new ideal functionality for AKE is adapted from $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ [41] which extends the standard key exchange ideal functionality [17] with resistance to Key-Compromise Impersonation (KCI) attacks (or KCI security for short).

Security against Key Control. In $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$, the adversary is allowed to control the session key of an uncompleted session between parties P and P', denoted by $\langle ssid, \mathsf{P}, \mathsf{P}' \rangle$, when (i) $\langle ssid, \mathsf{P}, \mathsf{P}' \rangle$ is Compromised, or (ii) P or P' is corrupted. In particular, a fresh session $\langle ssid, \mathsf{P}, \mathsf{P}' \rangle$ is marked Compromised when

the adversary takes an IMPERSONATE action on a COMPROMISED P'. The COMPROMISED action on P' models the leakage of P''s secret keys to the adversary. In contrast to a COMPROMISED party, a corrupted party is controlled by the adversary. Since both an adversary that holds P''s secret keys and impersonates P' in session $\langle ssid, \mathsf{P}, \mathsf{P}' \rangle$ and a corrupted party could control the session key, $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ does not imply security against key control.

To capture security against key control, our AKE functionality only permits the adversary to control the session key when (i) a party is corrupted and the adversary impersonates (i.e., takes an IMPERSONATE action on) this party's counterpart, or (ii) both parties are corrupted, as in either case the adversary is able to obtain the session key before the end of the session.

Weak Forward Secrecy. $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ captures full forward secrecy in the sense that once a session $\langle ssid, \mathsf{P}, \mathsf{P}' \rangle$ is completed, the adversary can no longer obtain the session key by compromising the party P or P'. Weak forward secrecy is the same as full forward secrecy except that the adversary is passive during session execution. A standard way to build AKE protocols with full forward secrecy is to add key confirmation messages derived from the session key to an AKE protocol with weak forward secrecy. For example, the key confirmation messages in the HMQV-C protocol [42] are two MAC values (computed on 0 and 1) under the session key. Since the session key is unknown to the RFs, it is unlikely for the RFs to check and sanitize these messages to defend against subversion attacks. Therefore, we only consider weak forward secrecy.

2.2 Our Technology Roadmap

Our starting point is the implicitly authenticated key exchange protocol Π by Cohn-Gordon et al. [28]. Assume that the public parameters of Π include $g \in \mathbb{G}$, where $\mathbb{G}$ is a cyclic group of prime order p. In Π, two parties $\mathsf{C_A}$, $\mathsf{C_B}$ with long-term private/public key pairs $(a, A = g^a)$ and $(b, B = g^b)$ send $U = g^r$ and $V = g^s$ to each other, and the session key $k_{\mathsf{A}} = H(\mathsf{trans} \| V^a \| B^r \| V^r) = H(\mathsf{trans} \| A^s \| U^b \| U^s) = k_{\mathsf{B}}$ where H is a hash function and trans is a concatenation of the transcript (U, V), party identities $(\mathsf{C_A}, \mathsf{C_B})$ and long-term public keys (A, B). In particular, V^r, U^s are ephemeral-ephemeral DH values, and V^a, A^s, B^r, U^b are static-ephemeral DH values.

Modifying Static-Ephemeral DH Values. Note that the computation of the DH values V^a, U^b can be seen as a decapsulation of the ciphertexts V and U with the long-term secret keys a, b. It is reasonable to make the following changes to the protocol Π while maintaining the functionality. $\mathsf{C_A}(\mathsf{C_B})$ sends an extra ElGamal encryption $c_{\mathsf{B}} = (g^\beta, B^\beta K_{\mathsf{B}})(c_{\mathsf{A}} = (g^\alpha, A^\alpha K_{\mathsf{A}}))$ of a random key $K_{\mathsf{B}}(K_{\mathsf{A}})$ under the long-term public key $B(A)$ to each other, and the session key $k_{\mathsf{A}} = H(\mathsf{trans}' \| c_{\mathsf{A},2}/c_{\mathsf{A},1}^a \| K_{\mathsf{B}} \| V^r) = H(\mathsf{trans}' \| K_{\mathsf{A}} \| c_{\mathsf{B},2}/c_{\mathsf{B},1}^b \| U^s) = k_{\mathsf{B}}$. The transcript in trans' is $(U, c_{\mathsf{B}}, V, c_{\mathsf{A}})$. We remark that this modification ensures that U and V are only used for computing the ephemeral-ephemeral DH value, making it easier to extract the generic framework later.

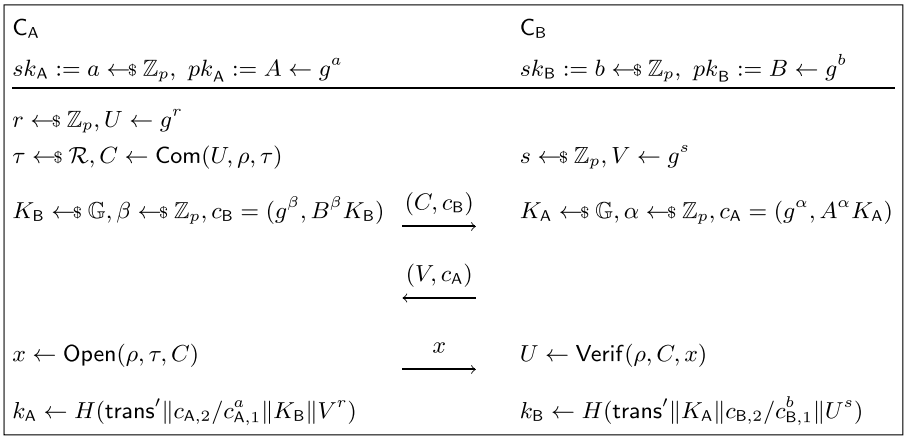

Fig. 1. The protocol Π' where $\mathsf{C_A}$ first sends the commitment of U and then opens it after receiving V from $\mathsf{C_B}$. Let $g \in \mathbb{G}, p = |\mathbb{G}|$ be public parameters. Let $\Pi_{\mathsf{com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ be the commitment scheme. $\rho \leftarrow\!\$\, \mathsf{KGen}(1^\lambda)$ represents the public parameters of Π_{com}, and $\mathcal{R}$ is the randomness space of Π_{com}.

Achieving Security against Key Control. Obviously, this modified protocol does not provide security against key control. After receiving (U, c_B) from $\mathsf{C_A}$, $\mathsf{C_B}$ can control the session key k by repeatedly sampling K_A or s until k meets certain requirements. To achieve security against key control, we follow the approach of Dodis et al. [30] and obtain a new protocol Π' shown in Fig. 1. That is, $\mathsf{C_A}$ commits to U and sends only the commitment of U and c_B to $\mathsf{C_B}$. Upon receiving (V, c_A) from $\mathsf{C_B}$, $\mathsf{C_A}$ opens the commitment to $\mathsf{C_B}$. In this case, $\mathsf{C_B}$ cannot derive the session key k until it receives the opening from $\mathsf{C_A}$, and loses control of k.

In Sect. 7, the commitment scheme depicted in Fig. 1 is instantiated as follows. The commitment to U is defined as $C = (g^y, h^y U)$, where $h \in \mathbb{G}$ is a public parameter and $y \leftarrow\!\$\, \mathbb{Z}_p$. The opening of the commitment C is a random value y^*. To verify the opening, one first parses $C = (C_1, C_2)$, checks whether $C_1 = g^y$ and then either outputs C_2/h^y or $\perp$, depending on the validity of C.

The First Attempt: Adding Reverse Firewalls. Next we consider inserting RFs for $\mathsf{C_A}$ and $\mathsf{C_B}$ to defend against subversion attacks. In particular, the RF $\mathsf{F_A}$ for $\mathsf{C_A}$ must rerandomize the outgoing ciphertext c_B and maul its underlying key K_B to K'_B using randomness Δ_B. Assume that the RF $\mathsf{F_B}$ for $\mathsf{C_B}$ does no further modification on this sanitized ciphertext, then $\mathsf{C_B}$ would retrieve K'_B from it. To maintain the functionality of the protocol Π', $\mathsf{F_A}$ must return Δ_B to $\mathsf{C_A}$ so that $\mathsf{C_A}$ can derive session key k_A with the same K'_B as $\mathsf{C_B}$. Similarly, $\mathsf{F_B}$ needs to rerandomize the ciphertext c_A, maul the key K_A to K'_A using the randomness Δ_A, and forward Δ_A back to $\mathsf{C_B}$. Clearly, both RFs are not transparent.

Furthermore, $\mathsf{F_B}$ is permitted only to rerandomize the incoming ciphertext c_B and should not maul the underlying K_B. If $\mathsf{F_B}$ were to alter K_B, it would be unable to return the randomness used in mauling K_B back to $\mathsf{C_A}$, causing both

parties to disagree on K_{B}. Likewise, F_{A} should not change the underlying K_{A} of the incoming ciphertext c_{A}. However, rerandomizing the incoming ciphertexts alone is not sufficient to defend against input-trigger attacks. In the unauthenticated channel setting, an adversary could send an encryption of a triggering message to the parties. Rerandomizing this encryption does not sanitize its malicious payload, and the whole protocol is vulnerable to input-trigger attacks.

C_{A}		C_{B}
$sk_{\mathsf{A}} := a \leftarrow\!\!\$\ \mathbb{Z}_p,\ pk_{\mathsf{A}} := A \leftarrow g^a$		$sk_{\mathsf{B}} := b \leftarrow\!\!\$\ \mathbb{Z}_p,\ pk_{\mathsf{B}} := B \leftarrow g^b$
$r \leftarrow\!\!\$\ \mathbb{Z}_p, U \leftarrow g^r$		
$\tau \leftarrow\!\!\$\ \mathcal{R}, C \leftarrow \mathsf{Com}(U, \rho, \tau)$		
$K_{\mathsf{B}} \leftarrow\!\!\$\ \mathbb{G}, \beta \leftarrow\!\!\$\ \mathbb{Z}_p, c_{\mathsf{B}} = (g^\beta, B^\beta K_{\mathsf{B}})$		$K_{\mathsf{A}} \leftarrow\!\!\$\ \mathbb{G}, V \leftarrow g^s$
$R_{\mathsf{A}} \leftarrow\!\!\$\ \mathbb{G}, \tau^* \leftarrow\!\!\$\ \mathcal{R}^*, \overline{C} \leftarrow \mathsf{Com}'(R_{\mathsf{A}}, \rho^*, \tau^*)$	$\xrightarrow{(C, c_{\mathsf{B}}, \overline{C})}$	$R_{\mathsf{B}} \leftarrow\!\!\$\ \mathbb{G}, c_{\mathsf{A}} = (g^\alpha, A^\alpha K_{\mathsf{A}})$
	$\xleftarrow{(V, c_{\mathsf{A}}, R_{\mathsf{B}})}$	
$x \leftarrow \mathsf{Open}(\rho, C, \tau),\ \overline{x} \leftarrow \mathsf{Open}'(\rho^*, \overline{C}, \tau^*)$	$\xrightarrow{(x, \overline{x})}$	$U \leftarrow \mathsf{Verif}(\rho, C, x)$
$R_{\mathsf{AB}} \leftarrow R_{\mathsf{A}} R_{\mathsf{B}}$		$R_{\mathsf{A}} \leftarrow \mathsf{Verif}'(\rho^*, \overline{C}, \overline{x})$
$\bar{K}_{\mathsf{A}} \leftarrow (c_{\mathsf{A},2}/c_{\mathsf{A},1}^a) R_{\mathsf{AB}}$		$\bar{K}_{\mathsf{B}} \leftarrow (c_{\mathsf{B},2}/c_{\mathsf{B},1}^b) R_{\mathsf{AB}}$
$k_{\mathsf{A}} \leftarrow H(\mathsf{aux} \Vert \bar{K}_{\mathsf{A}} \Vert K_{\mathsf{B}} R_{\mathsf{AB}} \Vert V^r)$		$k_{\mathsf{B}} \leftarrow H(\mathsf{aux} \Vert K_{\mathsf{A}} R_{\mathsf{AB}} \Vert \bar{K}_{\mathsf{B}} \Vert U^s)$

Fig. 2. The protocol Π'' is the same as Π' except that 1) C_{A} and C_{B} perform an extra coin tossing protocol, 2) the derivation of the session key involves the negotiated element R_{AB}, and 3) aux is identical to trans' but excludes the protocol transcript. Let $\Pi'_{\mathsf{Com}} = (\mathsf{KGen}', \mathsf{Com}', \mathsf{Open}', \mathsf{Verif}')$ be another commitment scheme, with $\rho^* \leftarrow\!\!\$\ \mathsf{KGen}'(1^\lambda)$ as its public parameters and $\mathcal{R}^*$ as its randomness space.

Our Treatment for Transparent RFs. To build an AKE protocol that is compatible with transparent RFs, which would also sanitize all incoming and outgoing messages for their cores, the protocol Π' needs an additional mechanism to ensure that both C_{A} and C_{B} would agree on the same K_{A} and K_{B}. As shown in Fig. 2, we ask C_{A} and C_{B} to negotiate an extra element R_{AB} by performing coin tossing, and this element allows C_{A} and C_{B} to reach the same session key. In specific, C_{A} first sends a commitment of a random R_{A}, then C_{B} returns a random R_{B}. Finally, C_{A} opens this commitment, and both parties get $R_{\mathsf{AB}} = R_{\mathsf{A}} R_{\mathsf{B}}$.

As the code for F_{A} and F_{B} is the same, we simplify the discussion by focusing solely on C_{A} having an RF F_{A}, as detailed in Fig. 3. Both commitment schemes are instantiated identically for brevity. In specific, F_{A} first mauls the underlying K_{B} of the ciphertext c_{B} to $K'_{\mathsf{B}} = K_{\mathsf{B}} \Delta_{\mathsf{B}}$ and the underlying U and R_{A} of the commitments C and $\overline{C}$ to $U' = U^{\Delta_r}$ and $R'_{\mathsf{A}} = R_{\mathsf{A}} \Delta_{\mathsf{A}} \Delta_{\mathsf{B}}^{-1}$, respectively. It then mauls the underlying K_{A} of the ciphertext c_{A} to $K'_{\mathsf{A}} = K_{\mathsf{A}} \Delta_{\mathsf{B}}^{-1}$, and modifies

V and R_{B} to $V' = V^{\Delta_r}$ and $R'_{\mathsf{B}} = R_{\mathsf{B}}\Delta_{\mathsf{A}}$. At last, $\mathsf{F_A}$ mauls the openings of commitments to maintain correctness. Now the session key derived by $\mathsf{C_A}$ is the same as the one by $\mathsf{C_B}$:

$$\begin{aligned}
k_{\mathsf{A}} &= H(\mathsf{aux}\|K'_{\mathsf{A}}R_{\mathsf{A}}R'_{\mathsf{B}}\|K_{\mathsf{B}}R_{\mathsf{A}}R'_{\mathsf{B}}\|V'^{r}) \\
&= H(\mathsf{aux}\|(K_{\mathsf{A}}\Delta_{\mathsf{B}}^{-1})R_{\mathsf{A}}(R_{\mathsf{B}}\Delta_{\mathsf{A}})\|K_{\mathsf{B}}R_{\mathsf{A}}(R_{\mathsf{B}}\Delta_{\mathsf{A}})\|(V^{\Delta_r})^{r}) \\
&= H(\mathsf{aux}\|K_{\mathsf{A}}R_{\mathsf{A}}R_{\mathsf{B}}\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}^{-1}\|K_{\mathsf{B}}R_{\mathsf{A}}R_{\mathsf{B}}\Delta_{\mathsf{A}}\|(V^{r})^{\Delta_r}) \\
&= H(\mathsf{aux}\|K_{\mathsf{A}}R_{\mathsf{A}}R_{\mathsf{B}}\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}^{-1}\|K_{\mathsf{B}}R_{\mathsf{A}}R_{\mathsf{B}}\Delta_{\mathsf{A}}(\Delta_{\mathsf{B}}\Delta_{\mathsf{B}}^{-1})\|(U^{s})^{\Delta_r}) \\
&= H(\mathsf{aux}\|K_{\mathsf{A}}(R_{\mathsf{A}}\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}^{-1})R_{\mathsf{B}}\|(K_{\mathsf{B}}\Delta_{\mathsf{B}})(R_{\mathsf{A}}\Delta_{\mathsf{A}}\Delta_{\mathsf{B}}^{-1})R_{\mathsf{B}}\|(U^{\Delta_r})^{s}) \\
&= H(\mathsf{aux}\|K_{\mathsf{A}}R'_{\mathsf{A}}R_{\mathsf{B}}\|K'_{\mathsf{B}}R'_{\mathsf{A}}R_{\mathsf{B}}\|U'^{s}) = k_{\mathsf{B}},
\end{aligned}$$

where aux does not include the transcript of protocol Π'' anymore, as the views of $\mathsf{C_A}$ and $\mathsf{C_B}$ on the transcript are different. We emphasize that removing the transcript from session key derivation allows an adversary to perform man-in-the-middle (MITM) attacks, manipulate messages like the RFs, and thereby impersonate an honest party. For protocols that are compatible with transparent RFs, such an impersonation attack is somewhat unavoidable [3], and applies to existing models of RFs [23,30]. Note that no sensitive information is leaked during an impersonation attack provided that the impersonated party is honest.

Generic Framework. Note that V can be viewed as an encapsulation of key U^s under the public key U. We generalize the protocol Π'' into an AKE framework based on KEM as shown in Fig. 4. Since $\mathsf{C_B}$ in Π'' has to send the KEM ciphertext V before receiving the public key U, the KEM ciphertext V shall be independent from the public key for the encapsulation algorithm. We call such a KEM *blind* and give the formal definition in Sect. 3.1.

In this generic framework of AKE protocol, the commitment schemes should be *malleable* and *rerandomizable* such that $\mathsf{F_A}$ could maul the underlying $\widetilde{pk}$ and R_{A} of the commitments C and $\overline{C}$ and rerandomize these commitments. Also, the blind KEM bKEM is required to be *key-malleable*, which means that its public key is malleable and a ciphertext under a mauled public key can be transformed into a ciphertext under the original public key. The regular KEM KEM is required to be *rerandomizable* and *ciphertext-malleable* (i.e., one can alter the encapsulated key by modifying the ciphertext). The algorithm KMaul is associated with the ciphertext malleability of KEM. To prove the protocol's security, we introduce a new security notion for blind KEM called *strong one-wayness*. Informally, given a blind KEM ciphertext c and a public key pk, it is hard to find (K, pk', c') such that $\mathsf{Decap}(sk, c') = \mathsf{Decap}(sk', c) = K$ where sk and sk' are the secret keys for pk and pk'. In particular, bKEM should satisfy this property.

In the same way that protocols by Chakraborty et al. [22,23] rely on the trusted setup assumption, we assume the generation of public parameters and long-term key pairs in this protocol does not suffer from subversion attacks. In fact, removing or weakening this trusted setup assumption is a non-trivial task, and has been left as an open problem in [23].

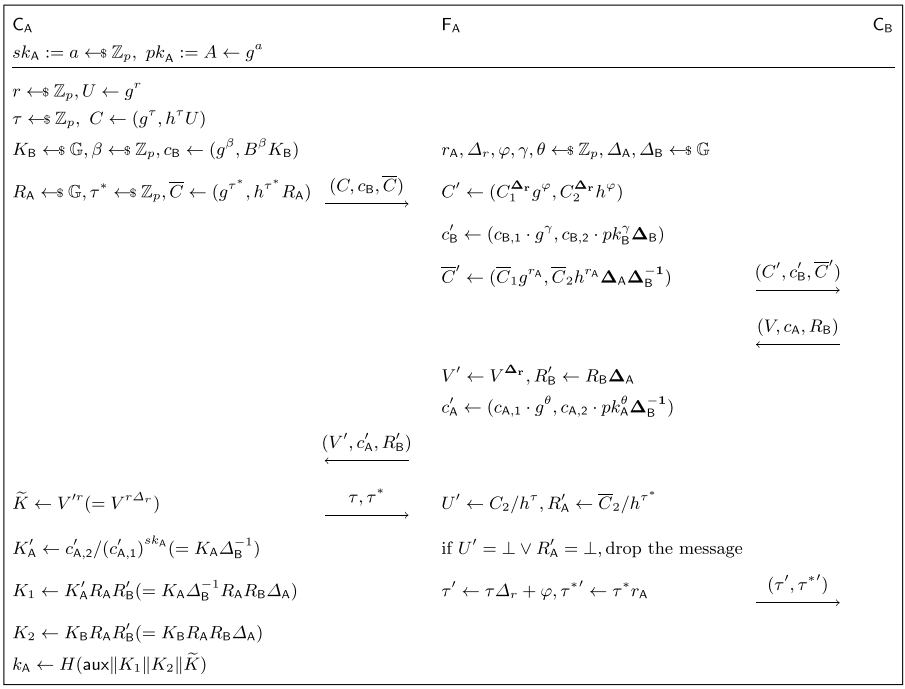

Fig. 3. The operations of $\mathsf{F_A}$, where $h \in \mathbb{G}$ is a public parameter for both commitment schemes. Let $C = (C_1, C_2), \overline{C} = (\overline{C}_1, \overline{C}_2)$. For a pair of values $X_r = (E, F)$, let $X_{r,1} := E, X_{r,2} := F$.

2.3 Related Work

Here we give an overview of previous countermeasures against subversion attacks. RFs have been used to protect various cryptographic schemes including signatures [4], message transmission protocols [30], zero-knowledge proofs [35], and general MPC protocols [18,19]. Existing AKE protocols with RFs [14,30,44] are proved in the standalone models. Specifically, Dodis et al. [30] proposed a mutually authenticated key agreement protocol that is compatible with transparent RFs. Bossuat et al. [14] presented RFs with public-private key pairs that allow sanitizing the message of the handshake protocol without requiring underlying primitives to be rerandomizable. Recently, Liu et al. [44] proposed a generic subversion-resilient AKE construction with RFs under the classical game-based model, and Liu et al. [46] introduced a modified TLS 1.3 handshake protocol with RFs following Arnold et al.'s model [2]. However, these two protocols do not consider input-trigger attacks and their RFs are not transparent. Moreover, Liu et al. [45] proposed a PAKE protocol with transparent RFs that UC-realizes a relaxed PAKE functionality under Arnold et al.'s model [2].

The self-guarding mechanism by Fischlin and Mazaheri [33] assumes a secure initialization phase of protocols where parties are not subverted, serving as "secu-

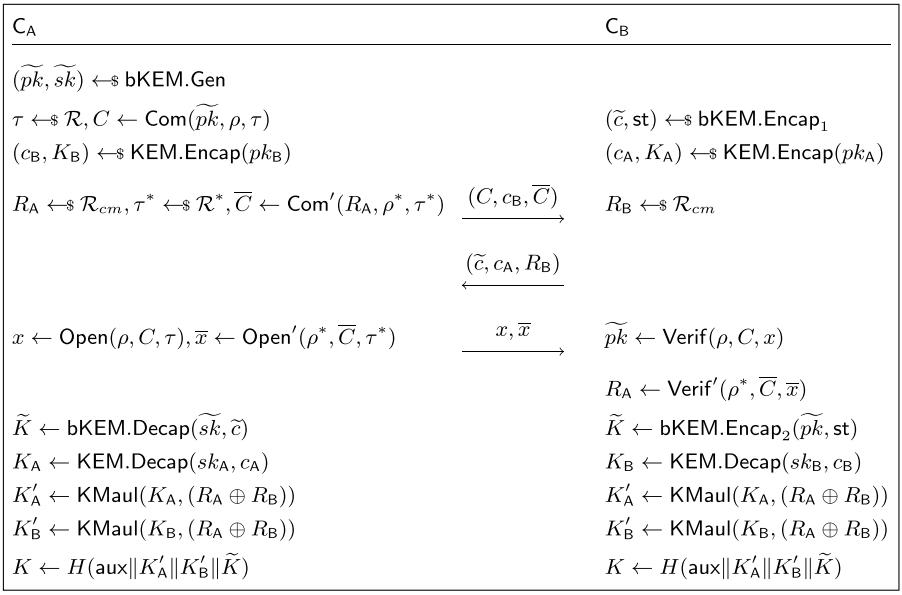

Fig. 4. A general framework from Π''. In particular, bKEM is a blind KEM whose encapsulation algorithm Encap can be split into two independent parts. The first part Encap_1 generates the KEM ciphertext without public key, and the second part Encap_2 derives the key from public key $\widetilde{pk}$ and states in Encap_1. pk_A and pk_B denote the long-term public keys for $\mathsf{C_A}$ and $\mathsf{C_B}$, and $\mathsf{aux} = \mathsf{C_A} \| \mathsf{C_B} \| pk_\mathsf{A} \| pk_\mathsf{B}$.

rity anchors". In comparison to RFs, it does not rely on an online third party. Moreover, a self-guarding mechanism is designed specifically to protect a key exchange protocol based on a physically unclonable function. Thus it could be a potential alternative to thwart subversions against AKE protocols.

Russell et al. [49] proposed the *watchdog* model to effectively detect the subversion attacks. This involves a detector "watchdog" that tests whether implementations of a cryptographic primitive strictly comply with its specification. Given the specification of a cryptographic scheme and its real-world implementation, a watchdog is allowed to detect whether the implementation is subverted. This method needs no trusted source of randomness (like the RFs) and is not limited by the samples collected during a clean initial phase (as in the self-guarding mechanism). A caveat is that it is challenging to resist the input-trigger attacks in this model. Over these years, watchdogs have been applied in many cryptographic primitives, such as one-way permutations [49], digital signatures [9,26], authenticated encryption with associated data [10], pseudorandom generator [49], public-key encryption [11,50], and random oracles [51].

3 Definitions and Components

Notations. Let g be a generator of the cyclic group $\mathbb{G}$ of prime order p. Let $y \leftarrow_{\$} \mathcal{Y}$ denote sampling a uniformly random element y from $\mathcal{Y}$. $x{:=}z$ means that the value of z is assigned to x. For a probabilistic algorithm $f(x)$ with input x, $y \leftarrow_{\$} f(x)$ represents a random output of $f(x)$ and $y \leftarrow f(x;r)$ represents the output of $f(x)$ using randomness r. For a positive integer N, $[N]$ denotes the integer set $\mathcal{X}{:=}\{1, 2, 3, \cdots, N\}$. Let λ denote the security parameter, $\mathsf{negl}(\lambda)$ denote a negligible function of λ and $p(\lambda)$ denote a polynomial of λ. For all polynomials $p(\lambda)$ and all sufficiently large λ, it holds that $\mathsf{negl}(\lambda) < \frac{1}{p(\lambda)}$.

3.1 Key Encapsulation Mechanism

A key encapsulation mechanism (KEM) [53] consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda)$ takes as input security parameter λ and produces public parameters pp, including the key space $\mathcal{K}$, public key space $\mathcal{PK}$, private key space $\mathcal{SK}$ and ciphertext space $\mathcal{C}$.
- $\mathsf{Gen}(\mathsf{pp})$ takes as input a public parameter pp and outputs a key pair (pk, sk).
- $\mathsf{Encap}(pk)$ takes as input a public key pk and outputs a ciphertext c and a key K.
- $\mathsf{Decap}(sk, c)$ takes as input a private key sk and a ciphertext c and outputs a key K.

Correctness. For all $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$, $(pk, sk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$ and $(c, K) \leftarrow_{\$} \mathsf{Encap}(pk)$, it holds that $\Pr[\mathsf{Decap}(sk, c) \neq K] \leq \mathsf{negl}(\lambda)$.

We define blindness for KEM as follows. In fact, the universal decryptability for KEM with no tags in [24] implies this property.

Definition 1. *(Blindness). For a KEM* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$*, we say* KEM *is blind if the* Encap *algorithm consists of following sub-algorithms.*

- *The randomness generation algorithm* $\mathsf{Rg}(\mathsf{pp})$*: takes as input a public parameter* pp*, outputs a randomness* r*.*
- *The key generation algorithm* $\mathsf{Kg}(pk, r)$*: takes as input a public key* pk *and a randomness* r*, outputs a key* K*.*
- *The ciphertext generation algorithm* $\mathsf{Cg}(r)$*: takes as input a randomness* r*, outputs a ciphertext* c*.*

and for all $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$*,* $r \leftarrow_{\$} \mathsf{Rg}(\mathsf{pp})$*,* $c \leftarrow \mathsf{Cg}(r)$ *and* $(pk, sk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$*, we have* $\Pr[\mathsf{Decap}(sk, c) \neq \mathsf{Kg}(pk, r)] \leq \mathsf{negl}(\lambda)$.

The malleability of the encapsulated key and the public key in a ciphertext is defined as follows. In particular, the definition of key malleability is adapted from the definition by Dodis et al. [30] for public-key encryption.

Definition 2. *(Ciphertext malleability). For a KEM* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$*, we say* KEM *is ciphertext-malleable if following conditions hold.*

- *There exists a pair of algorithms* $(\mathsf{CMaul}, \mathsf{KMaul})$ *as below.*

- $\mathsf{CMaul}(c, r)$*: takes as input a ciphertext* c *and a randomness* $r \in \mathcal{R}_{cm}$*, where* $\mathcal{R}_{cm}$ *is the randomness space, and outputs a new ciphertext* c'*;*
- $\mathsf{KMaul}(K, r)$*: takes as input a key* K *and a randomness* $r \in \mathcal{R}_{cm}$*, and outputs a new key* K'*;*

– *For* $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$*,* $(pk, sk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$*,* $(c, K) \leftarrow_{\$} \mathsf{Encap}(pk)$ *and any* $r \in \mathcal{R}_{cm}$*, we have* $\Pr[\mathsf{Decap}(sk, \mathsf{CMaul}(c, r)) \neq \mathsf{KMaul}(K, r)] \leq \mathsf{negl}(\lambda)$*;*

– *Let* $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$*,* $(pk, sk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$ *and* $K_0 \in \mathcal{K}$*. For* $(c, K) \leftarrow_{\$} \mathsf{Encap}(pk)$*,* $r \leftarrow_{\$} \mathcal{R}_{cm}$*,* $K' \leftarrow \mathsf{KMaul}(K_0, r)$*, the distributions of* K *and* K' *are identical.*

Definition 3. *(Key malleability). For a KEM* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$*, we say* KEM *is key-malleable if following conditions hold.*

– *The key pair* (pk, sk) *output by* Gen *is uniformly distributed over the space of valid key pairs.*

– *For a public key* pk*, there is a unique corresponding private key* sk*;*

– *There exists a pair of algorithms* $(\mathsf{CKeyMaul}, \mathsf{KeyMaul})$ *as below.*

- $\mathsf{CKeyMaul}(c, r)$*: takes as input a ciphertext* c *and a randomness* $r \in \mathcal{R}_{km}$*, and outputs a new ciphertext* c'*;*
- $\mathsf{KeyMaul}(pk, r)$*: takes as input a public key* pk *and a randomness* $r \in \mathcal{R}_{km}$*, where* $\mathcal{R}_{km}$ *is the randomness space, outputs a new public key* pk'*;*

– *Let* (pk, sk) *and* (pk', sk') *be two public/private key pairs such that* $pk' = \mathsf{KeyMaul}(pk, r)$ *for* $r \in \mathcal{R}_{km}$*. For* $(c, K) \leftarrow_{\$} \mathsf{Encap}(pk')$*, we have* $\Pr[\mathsf{Decap}(sk, \mathsf{CKeyMaul}(c, r)) \neq \mathsf{Decap}(sk', c)] \leq \mathsf{negl}(\lambda)$*;*

– *For any* $pk \in \mathcal{PK}$*,* $r \leftarrow_{\$} \mathcal{R}_{km}$*, we have* $pk' \leftarrow \mathsf{KeyMaul}(pk, r)$ *is uniformly distributed over the public key space* $\mathcal{PK}$*.*

Also, we adapt the definition by Faonio et al. [32] for public-key encryption, and define perfect rerandomizability for KEM as follows.

Definition 4. *(Perfect rerandomizability). For a KEM* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ *is perfectly rerandomizable if for any* $\mathsf{pp} \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$ *and* $(pk, sk) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$*, the following conditions hold:*

– *There exists an algorithm* $\mathsf{Rer}(c, r)$ *that takes as input a ciphertext* c *and a randomness* $r \in \mathcal{R}_{re}$*, where* $\mathcal{R}_{re}$ *is the randomness space, and outputs a new ciphertext* c'*;*

– *Given a ciphertext* c *s.t.* $(c, \cdot) \in \mathsf{Encap}(pk)$*, for any* $r \leftarrow_{\$} \mathcal{R}_{re}$*,* $c' \leftarrow \mathsf{Rer}(c, r)$ *and* $(c_0, \cdot) \leftarrow_{\$} \mathsf{Encap}(pk)$*, the distributions of* c_0 *and* c' *are identical.*

– *For any* $r \leftarrow_{\$} \mathcal{R}_{re}$*, ciphertexts* c *and* $c' \leftarrow \mathsf{Rer}(c, r)$*, it holds that* $\mathsf{Decap}(sk, c') = \mathsf{Decap}(sk, c)$*.*

– *For any (computationally unbounded) adversary* $\mathcal{A}$*,*

$$\Pr[(c, \cdot) \notin \mathsf{Encap}(pk) \wedge \mathsf{Decap}(sk, c) \neq \perp | c \leftarrow \mathcal{A}(pk)] \leq \mathsf{negl}(\lambda).$$

The first condition, combined with the last one, indicates that if a (potentially maliciously generated) ciphertext decrypts correctly, the distribution of a rerandomized ciphertext is statistically close to that of a fresh ciphertext [32].

We recall a practical security notion of KEM, called *one-wayness under plaintext checking attacks* (OW-PCA) [54] as below.

Definition 5. *(One-wayness under plaintext checking attacks* [54]**).** *For a KEM* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$*, we say* KEM *satisfies one-wayness under plaintext checking attacks (OW-PCA) if for any PPT adversary* $\mathcal{A}$*,* $\mathrm{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A}, \lambda) := \Pr[\mathsf{OW\text{-}PCA}(\mathcal{A}, \lambda, \mathsf{KEM}) = 1] \leq \mathsf{negl}(\lambda)$*, where game* $\mathsf{OW\text{-}PCA}(\mathcal{A}, \lambda, \mathsf{KEM})$ *is defined in Fig. 5.*

Exp $\mathsf{OW\text{-}PCA}(\mathcal{A}, \lambda, \mathsf{KEM})$	$\mathsf{PCO}_{\mathsf{KEM},sk}(c, K)$
1 : $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$	1 : $K^* \leftarrow \mathsf{Decap}(sk, c)$
2 : $(pk, sk) \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp})$	2 : **return** $K == K^*$
3 : $(c, K) \leftarrow\!\!\$\ \mathsf{Encap}(pk)$	
4 : $K' \leftarrow \mathcal{A}^{\mathsf{PCO}_{\mathsf{KEM},sk}(\cdot,\cdot)}(pk, c)$	
5 : **return** $K == K'$	

Fig. 5. The OW-PCA game for $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$.

We define a new notion called strong one-wayness for blind KEM as follows.

Definition 6. *(Strong one-wayness) For a blind KEM* $\mathsf{bKEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ *where* $\mathsf{Encap} = (\mathsf{Rg}, \mathsf{Kg}, \mathsf{Cg})$*, let* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$*,* $(pk, sk) \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp}), r \leftarrow\!\!\$\ \mathsf{Rg}(\mathsf{pp})$ *and* $c \leftarrow \mathsf{Cg}(r)$*, we say* bKEM *satisfies strong one-wayness if for any PPT adversary* $\mathcal{A}$*, it holds that* $\mathrm{Adv}^{\mathsf{stOW}}_{\mathsf{bKEM}}(\mathcal{A}, \lambda) := \Pr[K = \mathsf{Kg}(pk', r) = \mathsf{Decap}(sk, c') | (K, pk', c') \leftarrow \mathcal{A}(pk, c)] \leq \mathsf{negl}(\lambda)$.

3.2 Commitment Scheme

A commitment scheme consists of the following algorithms:

- $\mathsf{KGen}(1^\lambda)$ takes as input security parameter λ, and outputs the public parameter ρ, which includes the message space $\mathcal{M}_{com}$, randomness space $\mathcal{R}_{com}$, commitment space $\mathcal{C}_{com}$ and possibly the common reference string (CRS).
- $\mathsf{Com}(m, \rho, r)$ takes as input a message $m \in \mathcal{M}_{com}$, the public parameter ρ and randomness $r \in \mathcal{R}_{com}$, and outputs a commitment $C \in \mathcal{C}_{com}$.
- $\mathsf{Open}(\rho, C, r)$ takes as input the public parameter ρ, commitment C and randomness r, and outputs an opening x.
- $\mathsf{Verif}(\rho, C, x)$ takes as input the public parameter ρ, a commitment C and opening x, and outputs a message m or $\perp$.

Correctness. For all $\rho \leftarrow_{\$} \mathsf{KGen}(1^{\lambda}), m \in \mathcal{M}_{com}, r \leftarrow_{\$} \mathcal{R}_{com}, C \leftarrow \mathsf{Com}(m, \rho, r)$ and $x \leftarrow \mathsf{Open}(\rho, C, r)$, it holds that $\Pr[\mathsf{Verif}(\rho, C, x) \neq m] \leq \mathsf{negl}(\lambda)$.

As a basic security requirement, a commitment scheme should be binding and hiding. The formal definitions of these two properties are given in the full version.

We recall the definition of tightness for commitment as below.

Definition 7. (Tightness [30]) *A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ *is tight if for any* $\rho \leftarrow_{\$} \mathsf{KGen}(1^{\lambda})$*, commitment* C *and message* m*, there exists only one opening* x *such that* $\mathsf{Verif}(\rho, C, x) = m$.

We define the malleability and rerandomizability for a commitment scheme as follows. In particular, the definition of rerandomizability follows that in [30], while the definition of malleability does not require the plaintext space to be a group of order p, extending that in [30].

Definition 8. *(Malleability). Let* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ *be a commitment scheme. Let* $\mathcal{R}_{mcom}$ *be the randomness space for mauling. We say* Π_{Com} *is malleable if there exist three efficient algorithms* $(\mathsf{MaulCom}, \mathsf{mMaul}, \mathsf{xMaul})$ *such that for* $\rho \leftarrow_{\$} \mathsf{KGen}(1^{\lambda})$*, any commitment* $C \in \mathcal{C}_{com}$ *and its opening* x*, let* $r \leftarrow_{\$} \mathcal{R}_{mcom}$*,* $m' \leftarrow \mathsf{mMaul}(\mathsf{Verif}(\rho, C, x), r)$*,* $C' = \mathsf{MaulCom}(C, r)$ *and* $x' \leftarrow \mathsf{xMaul}(x, r)$*, it holds that* $m' = \mathsf{Verif}(\rho, C', x')$*. Moreover,* m' *is uniformly distributed over* $\mathcal{M}_{com}$.

Definition 9. *(Rerandomizability* [30]**).** *A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ *is rerandomizable if there exist two efficient algorithms* $(\mathsf{Rerand}, \mathsf{OpenRerand})$*, such that*

- *for any commitment* $C \in \mathcal{C}_{com}$ *and* $r \leftarrow_{\$} \mathcal{R}_{com}$*,* $C' \leftarrow \mathsf{Rerand}(C, r)$ *is a uniformly random value in commitment space; and*
- *for* $\rho \leftarrow_{\$} \mathsf{KGen}(1^{\lambda})$*, any commitment* $C \in \mathcal{C}_{com}$ *and opening* x*, it holds that* $\mathsf{Verif}(\rho, C, x) = \mathsf{Verif}(\rho, C', x')$ *and* $x' \leftarrow \mathsf{OpenRerand}(x, r)$.

To prove the UC security of the proposed AKE, we require the underlying commitments to be extractable, which implies the perfect binding property [23].

Definition 10. *(Extractability) A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ *is extractable if following conditions hold:*

- *The* KGen *algorithm outputs both the public parameter* ρ *and an extraction key* ek*, and there exists an efficient algorithm* Extract *that takes as input a commitment* C *and an extraction key* ek *and outputs a message* m*;*
- *For* $(\rho, ek) \leftarrow_{\$} \mathsf{KGen}(1^{\lambda})$*, any* $m \in \mathcal{M}_{com}$*,* $r \in \mathcal{R}_{com}$*,* $C \leftarrow \mathsf{Com}(m, \rho, r)$ *and* $x \leftarrow \mathsf{Open}(\rho, C, r)$*,* $\Pr[\mathsf{Verif}(\rho, C, x) = m \wedge \mathsf{Extract}(ek, C) \neq m] \leq \mathsf{negl}(\lambda)$.

4 The srUC Model

In this section, we present the srUC model in the setting of unauthenticated communication. In particular, we follow Canetti and Krawczyk's approach [17] that casts the *unauthenticated-links model* (UM) by Bellare et al. [6] in the UC model. That is, the UM is cast as a "hybrid model" with restricted access to the ideal authenticated communication functionality $\mathcal{F}_{\text{AUTH}}$[1] [15]. For each ordered pair of parties, access to $\mathcal{F}_{\text{AUTH}}$ is restricted to a single use throughout the computation. In other words, each party can only send one ideally authenticated message to every other party. Typically, this message includes initial information for establishing a long-term authentication mechanism, which corresponds to the "initialization function" [6,17] that is executed before the initiation of a protocol and securely generates the long-term information for parties. During subsequent executions, the adversary takes full control of the communication channel and can freely modify messages transmitted between RFs. We stress that all protocols presented in this work are described within the UM. Moreover, given that an RF is typically installed on the same machine as the core, it is reasonable to assume that communication between a core and its RF is secure.

Remark. Since all the protocols in [23] are depicted in the $\mathcal{F}_{\mathsf{SAT}}$-hybrid model, where $\mathcal{F}_{\mathsf{SAT}}$ denotes the ideal functionality for sanitizable authenticated communication, the definitions of the properties for RFs in the srUC model and the associated lemma (i.e., Lemma 1 in [23]) are also described in the setting of $\mathcal{F}_{\mathsf{SAT}}$-hybrid model. For AKE protocols, which run over unauthenticated channels, we need to tweak these definitions and the lemma in the srUC model.

4.1 Corruption Cases

In the srUC model, each party consists of a core and an RF. The core holds the secret inputs of the party and generates the messages of the protocol, while the RF does not have access to the secrets of the core and sanitizes the incoming/outgoing messages for the core with randomness. A core could be honest, specious or malicious, and an RF can be honest, semi-honest or malicious. The so-called specious corruption on a core only captures a specific type of subversion attacks that would not be detected via black-box test (see the full version for the formal description of specious corruption), as it is unlikely to achieve any security in the presence of arbitrary subversion attacks for certain ideal functionalities [23].

The combination of corruptions on core and RF results in 9 corruption cases which give rise to an honest, isolate or malicious party, as shown in Table 2. Each cell represents a corruption state of the party under the corresponding corruption

[1] $\mathcal{F}_{\text{AUTH}}$ guarantees authenticity of the communication content. Informally, $\mathcal{F}_{\text{AUTH}}$ operates by receiving a message $(\text{SEND}, sid, \mathcal{P}', m)$ from party $\mathcal{P}$, sending $(\text{SENT}, sid, \mathcal{P}, \mathcal{P}', m)$ to both $\mathcal{P}'$ and the adversary. Note that the adversary may delay the message sent to $\mathcal{P}'$, or determine its content and receiver identity (if undelivered) via a corruption request $(\text{CORRUPT}, sid, m', \mathcal{P}'')$.

case of the core and RF. For instance, if a core is speciously corrupted and an RF is honest, then the RF would faithfully sanitize messages for the core and the whole party can be viewed as an honest one. As a special case, when a core is honest and an RF is malicious, the composed party is isolated. The standard behavior of an ideal functionality $\mathcal{F}$ upon an isolate corruption is to perform a malicious corruption of the party in $\mathcal{F}$. However, as mentioned by Chakraborty et al. [22], in the unauthenticated communication setting, an isolate corruption is treated as part of the active adversary and therefore can be safely removed from the security analysis. For corruption cases that result in the same corruption on a party, one only needs to consider the simulation of the case (highlighted in Table 2) that gives the adversary the most capabilities.

Table 2. Corruption patterns in the real world and their translation to those of a party in the ideal world [23].

Core \ RF	Honest	SemiHonest	Malicious
Honest	Honest	Honest	Isolate
Specious	Honest	Malicious	Malicious
Malicious	Malicious	Malicious	Malicious

4.2 Wrapped Subversion-Resilient UC Security

In the srUC model, there are two types of ideal functionalities: sanitizable and regular ideal functionalities. The former provides explicit interfaces for the cores and RFs, while the latter are essentially ideal functionalities as in the UC model, interacting only with parties without being aware of cores and RFs.

To establish a structured connection between a regular ideal functionality $\mathcal{F}$ and cores/RFs, Chakraborty et al. [23] introduced the concept of a *wrapper* (denoted as Wrap) and *wrapped ideal functionality* $\mathsf{Wrap}(\mathcal{F})$. It runs the regular ideal functionality $\mathcal{F}$ internally, which is denoted by $\mathsf{Wrap}(\mathcal{F})$. As shown in Fig. 6, Wrap is used to provide the same interfaces for cores and RFs as in the protocol, forward the input of a core to the interface for the party on $\mathcal{F}$, and return the party's output to the core. Also, it translates corruptions of the core and RF into a corruption of the party following the corruption translation table (i.e., Table 2). The definition of UC security with subversion resilience for a regular ideal functionality is shown below, where specious environments/adversaries are formally defined in the full version. Roughly, in the srUC model, a specious environment would prepare a list of specious corruptions for the adversary, and a specious adversary would do specious corruptions in the list over the cores. Note that all specious corruptions in the srUC model are assumed to be static.

Definition 11. *(Wrapped subversion-resilient UC security [23]) Let $\mathcal{F}$ be an ideal functionality for n parties $\mathsf{P}_1, \cdots, \mathsf{P}_n$. Let Π be a sanitizing protocol with n cores $\mathsf{C}_1, \cdots, \mathsf{C}_n$ and n firewalls $\mathsf{F}_1, \cdots, \mathsf{F}_n$. Let $\mathcal{G}$ be a sanitizable ideal functionality that can be used by Π. We say that Π **wsrUC-realizes** $\mathcal{F}$ in the $\mathcal{G}$-hybrid model if Π UC-realizes $\mathsf{Wrap}(\mathcal{F})$ in the $\mathcal{G}$-hybrid model with the restriction that we only consider specious environments and specious adversaries.*

4.3 Properties of Reverse Firewalls

An RF [47] is initially defined as a stateful entity that maintains a list of session identifiers and associated state information. In particular, the former is used to identify different sessions and the latter is required to sanitize the corresponding session correctly.

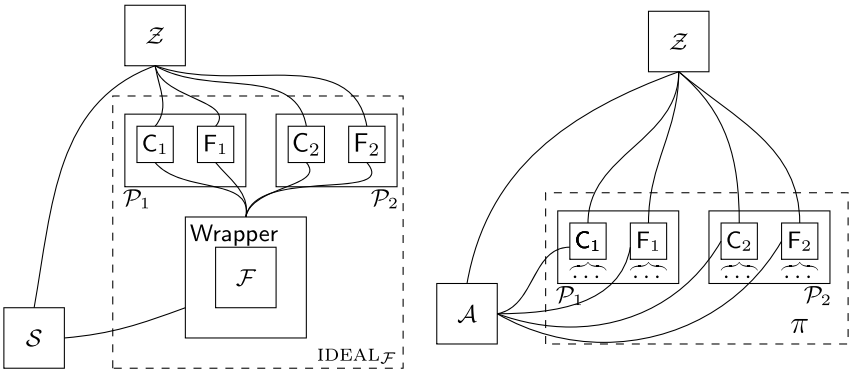

Fig. 6. The execution of an ideal protocol IDEAL$_\mathcal{F}$ (left) and a protocol π (right) in the srUC model. The wrapper in IDEAL$_\mathcal{F}$ forwards the input of C_i to the interface for $\mathcal{P}_i$ on $\mathcal{F}$, and returns the output of $\mathcal{P}_i$ to C_i. Also, the wrapper is responsible for translating corruptions on C_i and F_i into a corruption on $\mathcal{P}_i$. The cores and RFs in protocol π may have subroutines denoted by "$\cdots$".

One central property of RFs is strong sanitation. It ensures that no environment capable of performing specious corruptions on an honest core in the real world can distinguish between a protocol execution from one where the honest core is replaced by an incorruptible core (which remains honest during specious corruption), provided the honest core's firewall is also honest, i.e., the combination of a specious core and an honest RF is indistinguishable from the combination of an honest core and an honest RF. Let $\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, r)$ represent the random variable that describes the output of executing a protocol Π with the environment $\mathcal{Z}$ and adversary $\mathcal{A}$, using λ as the security parameter and r as input. Let $\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, r)\}_{r\in\{0,1\}^*}$ [15]. The formal definition of indistinguishable distribution ensembles is provided in the full version. The definition of strong sanitation below is the same as in [23] except that the protocol runs in the UM, rather than the $\mathcal{F}_{\mathsf{SAT}}$-hybrid model.

Definition 12. (Strong sanitation) *Let* $\mathcal{P} = (\mathsf{C}, \mathsf{F})$ *denote a party with a core* C *and an honest RF* F. *Let* $\widehat{\mathsf{C}}$ *denote the corresponding incorruptible core that would ignore any specious corruption. Let* Π_i *be a protocol running in the* UM *with the i-th party* $\mathcal{P}_i = (\mathsf{C}, \mathsf{F})$, *and all other parties are dummy. Let* Π_i' *be the protocol identical to* Π_i, *except for the i-th party* $\mathcal{P}_i = (\widehat{\mathsf{C}}, \mathsf{F})$. F *is* strongly sanitizing *if, for all PPT environments* $\mathcal{Z}$ *which are allowed to do specious corruptions of* C, *it holds that* $\text{EXEC}_{\Pi_i,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\Pi_i',\mathcal{A},\mathcal{Z}}$, *where* $\mathcal{A}$ *is the dummy adversary.*

Another important property of RFs is computational transparency. It captures the fact that an honest core without RF is indistinguishable from an honest core with RF. In [23], the definition of this property is limited to protocols in the $\mathcal{F}_{\mathsf{SAT}}$-hybrid model, as there is no complex interaction between $\mathcal{F}_{\mathsf{SAT}}$ and the RF, and it is feasible to replace an RF with an identity RF that only forwards messages for the core. Similar statements apply to the protocol in the UM, and we present the definition of computational transparency as follows.

Definition 13. (Computational transparency) *Let* ID *be an "identity" RF which only forwards messages for the core. Let* Π_i *be a protocol running in the* UM *with the i-th party* $\mathcal{P}_i = (\mathsf{C}, \mathsf{F})$, *and all other parties are dummy. Let* Π_i' *be the same as* Π_i *except that the i-th party is* $\mathcal{P}_i' = (\mathsf{C}, \mathsf{ID})$. *We say that* F *is* computationally transparent *if, for all PPT environments* $\mathcal{Z}$ *that do not corrupt* C *or* F/ID, *we have* $\text{EXEC}_{\Pi_i,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\Pi_i',\mathcal{A},\mathcal{Z}}$, *where* $\mathcal{A}$ *is a dummy adversary.*

Using the above two properties of RFs, the corruption cases requiring simulation can be further reduced or weakened, as per Lemma 2 in [23]. We restate this lemma in the UM as follows, while treating isolate corruptions as MITM attacks. Its proof is provided in the full version, which is similar to that in [23].

Lemma 1. *Let* Π *be a protocol of n parties* $(\mathsf{C}_i, \mathsf{F}_i)$ $(i = 1, 2, \cdots, n)$. *Assume that* Π *is executed in the* UM, *with a secure channel between each core and its RF. If all the RFs in* Π *are strongly sanitizing, then it suffices to consider the following corruption cases: (1) Honest core & semi-honest RF. (2) Malicious core & malicious RF. Moreover, if all the RFs are computationally transparent, then it suffices to consider the following corruption cases: (1) Honest core & Honest RF. (2) Malicious core & malicious RF.*

5 Ideal Functionality for AKE

Our ideal functionality $\mathcal{F}_{\mathsf{AKE}}$ for AKE is adapted from $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ by Jarecki et al. [41], both illustrated in Fig. 7. We first give a brief overview of $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$, and then explain the rationales behind the modifications on this ideal functionality.

Overview of $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$. First, $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ makes some syntactic adjustments of the original key exchange functionality $\mathcal{F}_{\mathsf{KE}}$ [17] to the user-server setting. The key exchange functionality has three basic interfaces: UsrSession, SrvSession, and NewKey. Users initiate an AKE session via UsrSession, while servers use

SrvSession. Both parties derive a session key by issuing a NewKey query. If both are honest, the functionality provides a uniform random key; otherwise, the adversary determines the key.

$\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ extends $\mathcal{F}_{\mathsf{KE}}$ by adding two adversarial behaviors: Compromise and Impersonate. Compromise models an adversary (referred to as $\mathcal{S}$) obtaining the private keys of a party. If $\mathcal{S}$ makes a (Compromise, $sid, \mathcal{P}$) query, then $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ marks $\mathcal{P}$ as compromised. $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ handles *adaptive* compromise by which $\mathcal{S}$ is allowed to make a Compromise query at any time (before, during or after a session execution). Compared with corrupting a party $\mathcal{P}$, $\mathcal{S}$ is not allowed to control a compromised $\mathcal{P}$. However, $\mathcal{S}$ can impersonate $\mathcal{P}$ in sessions with another party $\mathcal{P}'$, which is captured by the Impersonate interface..

A query (Impersonate, $sid, ssid, \mathcal{P}$) from $\mathcal{S}$ models the impersonation of $\mathcal{P}$ (that has been labeled compromised) in session $\langle ssid, \cdot, \mathcal{P}\rangle$, and $\mathcal{S}$ could determine the session key. If party $\mathcal{P}$ is labeled compromised but not corrupted, $\mathcal{P}$'s sessions that are uncorrupted and not labeled compromised are not considered compromised. This captures KCI security, i.e., an adversary that compromises a party $\mathcal{P}$ can impersonate $\mathcal{P}$, but can not impersonate other parties to $\mathcal{P}$.

Changes in the AKE functionality. For clarity, we choose to add the session key κ into a record $\langle ssid, \mathcal{P}, \mathcal{P}', \kappa\rangle$ instead of using the completed label. If a session key is not determined yet, we set $\kappa = \perp$.

There are two major modifications to $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$. The first one involves the cases where the adversary determines the session key. In $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$, the adversary determines the session key of $\mathcal{P}$ in session $\langle ssid, \mathcal{P}, \mathcal{P}'\rangle$ if (1) the record $\langle ssid, \mathcal{P}, \mathcal{P}', \perp\rangle$ is marked compromised, or (2) $\mathcal{P}$ or $\mathcal{P}'$ is corrupted. By comparison, in $\mathcal{F}_{\mathsf{AKE}}$, the adversary is only allowed to control the session key of party $\mathcal{P}$ in session $\langle ssid, \mathcal{P}, \mathcal{P}'\rangle$ if (1) $\mathcal{P}$ is corrupted and the record $\langle ssid, \mathcal{P}, \mathcal{P}', \perp\rangle$ is not marked fresh, or (2) both parties are corrupted. Specifically,

(i) The key of session $\langle ssid, \mathcal{P}, \mathcal{P}'\rangle$ is no longer determined by the adversary when (a) only $\mathcal{P}$ is corrupted and the record $\langle ssid, \mathcal{P}, \mathcal{P}', \perp\rangle$ is fresh (i.e., $\mathcal{S}$ does not send a (Impersonate, $sid, ssid, \mathcal{P}'$) query), or (b) only $\mathcal{P}'$ is corrupted, which captures security against key control. As shown by Dodis et al. [30], a corrupt implementation of either party has the potential to manipulate the value of session key without being detected. This change prevents any corrupted party interacting with an uncorrupted party from controlling the session key.

(ii) The adversary is only allowed to obtain, rather than determine, the session key of uncorrupted party $\mathcal{P}$ in a session $\langle ssid, \mathcal{P}, \mathcal{P}'\rangle$ whose record $\langle ssid, \mathcal{P}, \mathcal{P}', \cdot\rangle$ has been marked compromised. This change also captures security against key control: The compromised record $\langle ssid, \mathcal{P}, \mathcal{P}', \cdot\rangle$ implies that $\mathcal{P}'$ has been marked compromised and $\mathcal{S}$ has impersonated $\mathcal{P}'$ in session $\langle ssid, \mathcal{P}, \mathcal{P}'\rangle$. Therefore, $\mathcal{S}$ interacts with an uncorrupted party $\mathcal{P}$ in the name of $\mathcal{P}'$. The session key is unbiased due to security against key control. However, $\mathcal{S}$ is allowed to obtain the session key since it has impersonated $\mathcal{P}'$ by knowing the private key of $\mathcal{P}'$.

The second modification is related to weak forward secrecy. In $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$, the full forward secrecy is implicitly captured: the only behavior of $\mathcal{S}$ upon receiving

Suppose that $\mathcal{P}, \mathcal{P}' \in \{\mathsf{U}, \mathsf{S}\}$. Let $\mathcal{S}$ denote the adversary.
Upon receiving a query $(\textsc{UsrSession}, sid, ssid, \mathsf{S})$ from U:
- Send $(\textsc{UsrSession}, sid, ssid, \mathsf{U}, \mathsf{S})$ to $\mathcal{S}$.
- If $ssid$ was not used before by U, create session record $\langle ssid, \mathsf{U}, \mathsf{S}, \perp \rangle$ marked FRESH and send $(\textsc{UsrSession}, sid, ssid, \mathsf{U}, \mathsf{S})$ to $\mathcal{S}$.

Upon receiving a query $(\textsc{SrvSession}, sid, ssid, \mathsf{U})$ from S:
- Send $(\textsc{SrvSession}, sid, ssid, \mathsf{U}, \mathsf{S})$ to $\mathcal{S}$.
- If $ssid$ was not used before by S, create session record $\langle ssid, \mathsf{S}, \mathsf{U}, \perp \rangle$ marked FRESH and send $(\textsc{SrvSession}, sid, ssid, \mathsf{U}, \mathsf{S})$ to $\mathcal{S}$.

Upon receiving a query $(\textsc{Compromise}, sid, \mathcal{P})$ from $\mathcal{S}$:
- Mark $\mathcal{P}$ COMPROMISED.
- If there is a record $\langle ssid, \mathcal{P}', \mathcal{P}, \kappa \rangle$ marked INTERFERED, then send $(ssid, \mathcal{P}', \mathcal{P}, \kappa)$ to $\mathcal{S}$.

Upon receiving a query $(\textsc{Impersonate}, sid, ssid, \mathcal{P})$ from $\mathcal{S}$: If there is a record $\langle ssid, \mathcal{P}', \mathcal{P}, \perp \rangle$ marked FRESH:
- If $\mathcal{P}$ is marked COMPROMISED, mark this record COMPROMISED.
- Else mark this record INTERFERED.

Upon receiving a query $(\textsc{NewKey}, sid, ssid, \mathcal{P}, \kappa^*)$ from $\mathcal{S}$, where $|\kappa^*| = \lambda$, if there is a record $\mathsf{rec} = \langle ssid, \mathcal{P}, \mathcal{P}', \perp \rangle$, then:
- If $\mathcal{P}$ is corrupted and rec is not FRESH, or both $\mathcal{P}$ **and** $\mathcal{P}'$ are corrupted, set $\kappa \leftarrow \kappa^*$.
- If rec is COMPROMISED, or $\mathcal{P}$ or $\mathcal{P}'$ is corrupted, set $\kappa \leftarrow \kappa^*$.
- Else if rec is marked FRESH and there is a record $\langle ssid, \mathcal{P}', \mathcal{P}, \kappa' \rangle$ marked FRESH, set $\kappa \leftarrow \kappa'$.
- Else, pick $\kappa \leftarrow\!\!\$\ \mathcal{K}$, where $\mathcal{K}$ is the key space.

If rec is COMPROMISED, send κ to $\mathcal{S}$. Finally, update rec to $\langle ssid, \mathcal{P}, \mathcal{P}', \kappa \rangle$ and send $(sid, ssid, \kappa)$ to $\mathcal{P}$.

Fig. 7. AKE functionality $\mathcal{F}_{\mathsf{AKE}}$, which includes all codes except the boxed text. $\mathcal{F}_{\mathsf{AKE\text{-}KCI}}$ [41] includes all codes but the dark gray parts.

a query $(\textsc{Compromise}, sid, \mathcal{P})$ is to mark $\mathcal{P}$ as COMPROMISED, which means that once the session key κ has been generated, the COMPROMISE query does not help obtain κ. In $\mathcal{F}_{\mathsf{AKE}}$, weak forward secrecy is captured by strengthening the ability of $\mathcal{S}$. In specific, there is an additional check upon receiving a $(\textsc{Compromise}, sid, \mathcal{P})$ query: If there is a record $\langle ssid, \mathcal{P}', \mathcal{P}, \kappa \rangle$ not marked FRESH, then sends κ to $\mathcal{S}$. The rationale for this design is as below.

In general, full forward secrecy ensures the security of a session regardless of COMPROMISE of either party after the session is completed. For an AKE protocol with implicit authentication and weak forward secrecy, a standard way to achieve full forward secrecy is to add key confirmation messages (derived from the session key) [36]. However, only the parties who possess the session key can compute the key confirmation message correctly, which makes it difficult for an RF to check and sanitize the key confirmation messages. For better compatibility of AKE protocols with RFs, we aim to achieve weak forward secrecy.

Weak forward secrecy ensures the security of a passively observed session regardless of COMPROMISE of either party. If a session $\langle ssid, \mathcal{P}, \mathcal{P}' \rangle$ was actively interfered in, and $\mathcal{P}'$ is COMPROMISED, the security of session keys is not guaranteed. In $\mathcal{F}_{\mathsf{AKE}}$, the COMPROMISED and INTERFERED marks of a record model the ability of $\mathcal{S}$ to actively interfere in sessions, where a record $\langle ssid, \mathcal{P}', \mathcal{P}, \cdot \rangle$ is marked INTERFERED if $\mathcal{S}$ makes a (IMPERSONATE, $sid, ssid, \mathcal{P}$) query but $\mathcal{P}$ is not marked COMPROMISED. A session that is not marked COMPROMISED or INTERFERED is always considered to be FRESH. A record $\langle ssid, \mathcal{P}', \mathcal{P}, \kappa \rangle$ being not FRESH implies that it was actively interfered in before κ is generated. In such a case, COMPROMISE of $\mathcal{P}$ would reveal κ by weak forward secrecy. Note that $\mathcal{F}_{\mathsf{khAKE}}$ in [37] also captures weak forward secrecy through the Compromise and Interfere actions of the adversary.

6 Generic Construction

6.1 Building Blocks

Our generic subversion-resilient AKE protocol srAKE, is built on a blind KEM bKEM, a regular KEM KEM and two commitment schemes $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$, $\Pi'_{\mathsf{Com}} = (\mathsf{KGen}', \mathsf{Com}', \mathsf{Open}', \mathsf{Verif}')$. KEM satisfies OW-PCA, bKEM satisfies strong one-wayness, and both Π_{Com} and Π'_{Com} are computationally hiding, tight and extractable. To support the RFs, we require that bKEM is key-malleable, KEM is ciphertext-malleable, both Π_{Com} and Π'_{Com} are rerandomizable. Moreover, Π_{Com} is malleable with respect to bKEM, and Π'_{Com} is malleable with respect to KEM. In particular, let $\mathcal{PK}_{\mathsf{bKEM}}$ be the public key space of bKEM, $\mathcal{R}_{cm}$ the randomness space of KEM, and $\mathcal{M}_{com}$, $\mathcal{M}'_{com}$ the message spaces of Π_{Com} and Π'_{Com}, respectively. Then, $\mathcal{PK}_{\mathsf{bKEM}} \subseteq \mathcal{M}_{com}$ and $\mathcal{R}_{cm} \subseteq \mathcal{M}'_{com}$ are required. Below are the formal definitions.

Definition 14. (Malleability with respect to a key-malleable KEM) *Let* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ *be a commitment scheme and* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ *be a key-malleable KEM with algorithms* $(\mathsf{CKeyMaul}, \mathsf{KeyMaul})$. *We say* Π_{Com} *is malleable with respect to* KEM *if (1)* Π_{Com} *is malleable, i.e., there exist three algorithms* $(\mathsf{MaulCom}, \mathsf{mMaul}, \mathsf{xMaul})$*; and (2) for any pk generated by* Gen *and* $r \in \mathcal{R}_{mcom} \cap \mathcal{R}_{km}$*, where* $\mathcal{R}_{mcom}$ *is the randomness space for mauling in* Π_{Com} *and* $\mathcal{R}_{km}$ *is the randomness space of* KEM*, it holds that* $pk \in \mathcal{M}_{com}$ *and* $\mathsf{mMaul}(pk, r) = \mathsf{KeyMaul}(pk, r)$.

For $X \in \{\mathsf{CMaul}, \mathsf{KMaul}, \mathsf{CKeyMaul}, \mathsf{KeyMaul}, \mathsf{MaulCom}, \mathsf{MaulCom}'\}$, algorithm X takes as input an element and a randomness, and outputs a new element. Let $\mathcal{M}_{in}$ and $\mathcal{R}_m$ be the element space and randomness space respectively, and $X(c, r_1 \oplus r_2)$ denote $X(X(c, r_1), r_2)$. For all $c \in \mathcal{M}_{in}$, $r_1, r_2 \in \mathcal{R}_m$, it is required that $X(c, r_1 \oplus r_2) = X(c, r_2 \oplus r_1)$. For all $r \in \mathcal{R}_m$, we say r^{-1} is the *inverse element* of r if $X(c, (r \oplus r^{-1})) = c$.

Definition 15. (Malleability with respect to a ciphertext-malleable KEM) *Let* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verif})$ *be a commitment scheme and*

$\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ *be a ciphertext-malleable KEM with algorithms* $(\mathsf{CMaul}, \mathsf{KMaul})$. *We say* Π_{Com} *is malleable with respect to* KEM *if (1)* Π_{Com} *is malleable, i.e., there exist three algorithms* $(\mathsf{MaulCom}, \mathsf{mMaul}, \mathsf{xMaul})$*; and (2) it holds that* $\mathcal{R}_{cm} \subseteq \mathcal{M}_{com}$ *and* $\mathcal{R}_{mcom} = \mathcal{R}_{cm}$. *Moreover, for* $r, r_{cm} \in \mathcal{R}_{cm}$ *and any* K *generated by* Encap, *it holds that* $\mathsf{KMaul}(K, \mathsf{mMaul}(r_{cm}, r)) = \mathsf{KMaul}(K, r_{cm} \oplus r)$.

6.2 Protocol srAKE

Assume that srAKE is executed between two parties $\mathsf{U} = (\mathsf{C}_i, \mathsf{F}_i)$ and $\mathsf{S} = (\mathsf{C}_j, \mathsf{F}_j)$. The behaviors of U and S are respectively illustrated in Fig. 8 and Fig. 9, where $H : \{0,1\}^* \to \{0,1\}^w$ is a hash function with $w = p(\lambda)$. Note that the description of F_j is identical to that of F_i. The public parameters $\mathsf{pp} \leftarrow \mathsf{bKEM.Setup}(1^\lambda)$, $\mathsf{pp}' \leftarrow \mathsf{KEM.Setup}(1^\lambda)$, $\rho \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda), \rho' \leftarrow\!\!\$\ \mathsf{KGen}'(1^\lambda)$ are pre-shared by parties. At the beginning of the protocol, the cores C_i and C_j generate their long-term public/private key pairs (pk_i, sk_i) and (pk_j, sk_j) via

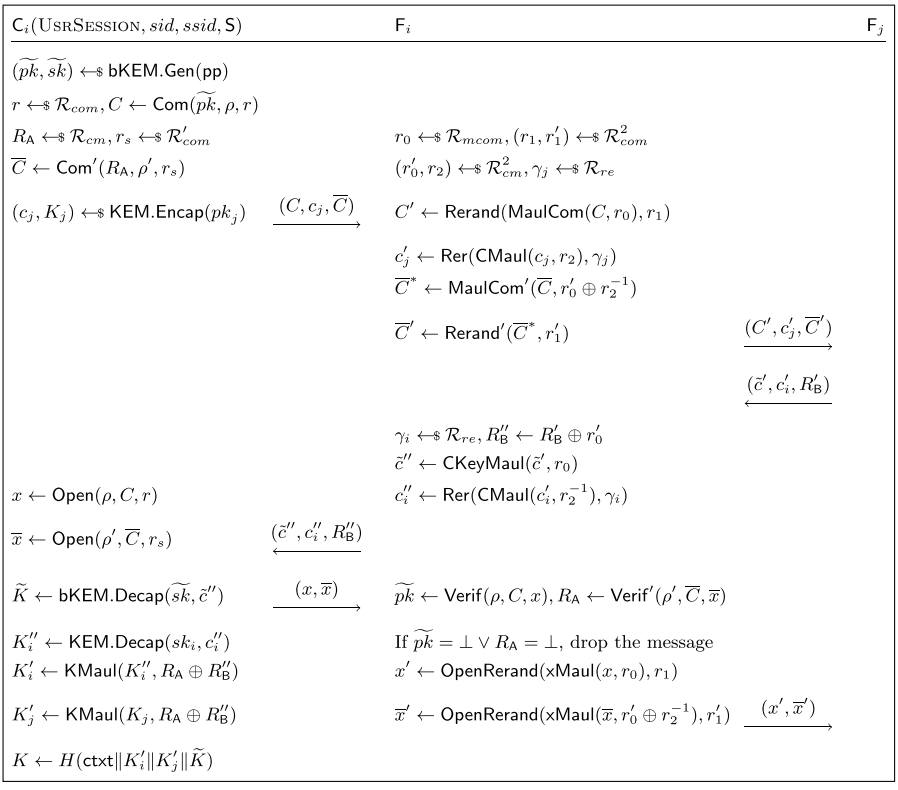

Fig. 8. C_i and F_i in srAKE. Let $\mathsf{ctxt} = sid \| ssid \| \mathsf{aux}$ where $\mathsf{aux} := \mathsf{C}_i \| \mathsf{C}_j \| pk_i \| pk_j$.

$\mathsf{KEM.Gen}(\mathsf{pp}')$, then send the public keys to each other using $\mathcal{F}_{\text{AUTH}}$. This initialization phase is omitted in Fig. 8 and Fig. 9. To adhere to the UC notion of AKE modeled by $\mathcal{F}_{\mathsf{AKE}}$, we incorporate the session identifier *sid* and sub-session identifier *ssid* into the session key derivation process, a practice commonly employed in UC-secure AKE protocols [37,41][2]. A textual representation of srAKE and the correctness analysis are shown in the full version.

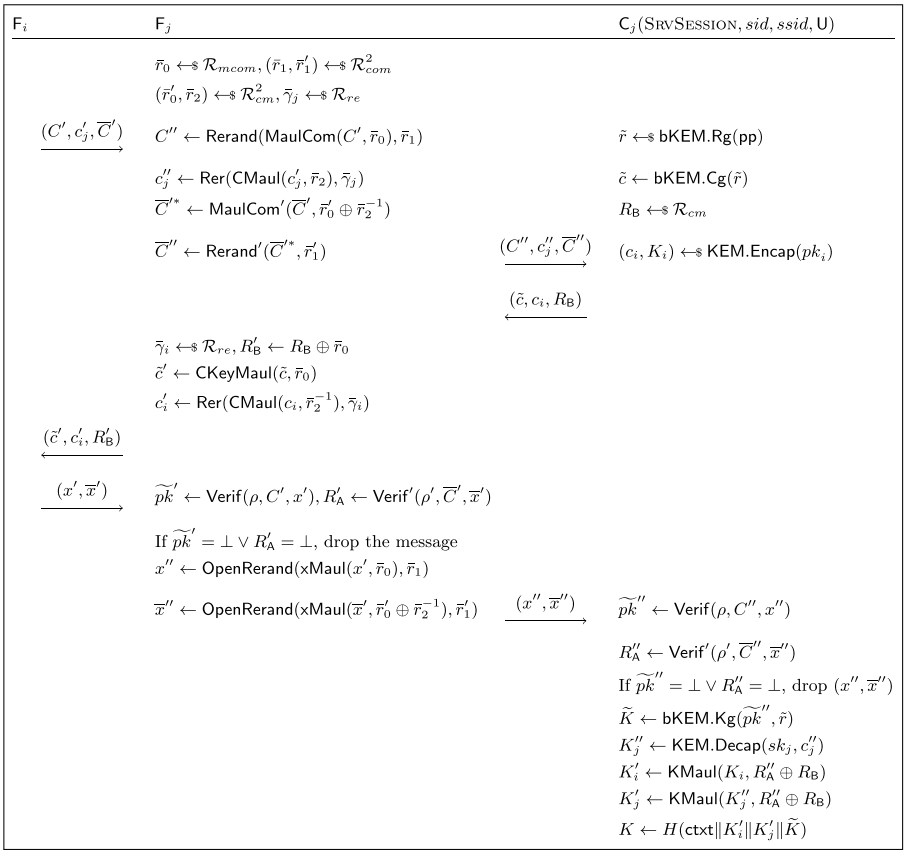

Fig. 9. F_j and C_j in srAKE.

Theorem 1. *Let* KEM *be a KEM,* bKEM *be a blind KEM,* Π_{Com} *and* Π'_{Com} *be two commitment schemes, all as defined above. The* srAKE *protocol wsrUC-realizes* $\mathcal{F}_{\mathsf{AKE}}$ *under static corruptions when* H *is modeled as an RO.*

[2] As Gu et al. [38] mentioned, in UC conventions, the top-level state of a specific cryptographic application, such as long-term keys of a public-key encryption, is called its "session", whereas the specific instances that utilize these long-term keys are termed "sub-sessions".

Proof. By the definition of wsrUC security, we prove that srAKE UC-realizes $\mathsf{Wrap}(\mathcal{F}_{\mathsf{AKE}})$. In particular, we analyze various corruption cases for both parties. Since our protocol operates over unauthenticated channels, the adversary has control over the network channel and can insert, alter, or discard messages.

To simplify the corruption cases, we prove the strong sanitation (Definition 12) and computational transparency (Definition 13) of the RFs in srAKE. Since the isolate corruption collapses to MITM attacks in the setting of unauthenticated communication [21], it is not considered explicitly in our security analysis. Subsequently, by Lemma 1, we consider two corruption types: *both the core and firewall are honest*, resulting in an honest party; *or both are malicious*, resulting in a malicious party. These lead to two corruption cases for analysis in srAKE: one malicious and one honest party; or both parties being honest.

Lemma 2. *The firewalls* F_i *and* F_j *are strongly sanitizing.*

Proof. To prove the strong sanitation of F_i, we claim that for all specious environments $\mathcal{Z}$ which are not allowed to corrupt F_i, but are permitted to speciously corrupt C_i, $\text{EXEC}_{\mathsf{srAKE},\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathsf{srAKE}^*,\mathcal{A},\mathcal{Z}}$ where srAKE and srAKE^* are executed by dummy parties except for U which is either $(\mathsf{C}_i, \mathsf{F}_i)$ or $(\widehat{\mathsf{C}}_i, \mathsf{F}_i)$ where $\widehat{\mathsf{C}}_i$ is an incorruptible core. If $\mathcal{Z}$ sends a specious corruption to C_i, then srAKE runs with a specious core $\widetilde{\mathsf{C}}_i$ while srAKE^* still runs with $\widehat{\mathsf{C}}_i$. Following Chakraborty et al. [21], we first demonstrate that an adversary cannot mount input-triggered attacks against C_i. We then prove that C_i reveals no information, i.e., the messages output by F_i remain unbiased.

For the incoming message $v'' := (\tilde{c}'', c_i'', R_{\mathsf{B}}'')$ of C_i: R_{B}'' is uniformly random due to the use of r_0'. By the key malleability of bKEM, it holds that $\widetilde{K} = \mathsf{bKEM.Decap}(\widetilde{sk}, \tilde{c}'') = \mathsf{bKEM.Decap}(\widetilde{sk}', \tilde{c}')$ where $\tilde{c}'' = \mathsf{CKeyMaul}(\tilde{c}', r_0)$ and $\widetilde{sk}'$ is the corresponding private key of $\widetilde{pk}'$. Since $\widetilde{pk}' = \mathsf{KeyMaul}(\widetilde{pk}, r_0)$ is uniformly random, we have that $\widetilde{K} = \mathsf{bKEM.Decap}(\widetilde{sk}', \tilde{c}')$ computed by C_i is uniformly random for a given $\tilde{c}'$. Therefore, $\tilde{c}''$ received by C_i is also uniformly random (for a given $\widetilde{sk}$). Let $(\hat{c}_i, \widehat{K}_i) \leftarrow\!\!\$\ \mathsf{Encap}(pk_i)$. The distribution of $c_i'' = \mathsf{Rer}(\mathsf{CMaul}(c_i', r_2^{-1}), \gamma_i)$ is the same as that of $\hat{c}_i$ by the perfect rerandomizability of KEM. Moreover, the distributions of the underlying key K_i'' of c_i'' and $\widehat{K}_i$ are identical by the ciphertext malleability of KEM. C_i gets $K_i' = \mathsf{KMaul}(K_i'', R_{\mathsf{A}} \oplus R_{\mathsf{B}}'')$, $K_j' = \mathsf{KMaul}(K_j, R_{\mathsf{A}} \oplus R_{\mathsf{B}}'')$, where $R_{\mathsf{A}} \oplus R_{\mathsf{B}}''$ is uniformly random. Therefore, K_i' and K_j' computed by C_i are unbiased.

Now we show that $u' = (C', c_j', \overline{C}')$ and $(x', \overline{x}')$ output by F_i are unbiased. Let $(\hat{c}_j, \widehat{K}_j) \leftarrow\!\!\$\ \mathsf{Encap}(pk_j)$. A specious C_i may send a biased $u = (C, c_j, \overline{C})$. However, this bias is always eliminated by an honest F_i. Upon receiving u from C_i, F_i sanitizes u using the $\mathsf{MaulCom}, \mathsf{Rerand}, \mathsf{MaulCom}', \mathsf{Rerand}', \mathsf{CMaul}$ and Rer algorithms. It is obvious that u' is unbiased: By the rerandomizability of Π_{Com} and Π_{Com}', C' and $\overline{C}'$ are uniformly random. By the perfect rerandomizability of KEM, c_j' has the same distribution as $\hat{c}_j$. Since Π_{Com} is malleable with respect to bKEM, it holds that $\widetilde{pk}' = \mathsf{mMaul}(\widetilde{pk}, r_0)$ is uniformly random over

$\mathcal{PK}_{\mathsf{bKEM}}$. Moreover, by the tightness of Π_{Com}, for commitment C' and message $\widetilde{pk}'$, the opening message $x' = \mathsf{OpenRerand}(\mathsf{xMaul}(x, r_0), r_1)$ is unique. x' is also uniformly random due to the uniform randomness of $\widetilde{pk}'$. Analogously, since Π'_{Com} is malleable, it holds that $R'_{\mathsf{A}} = \mathsf{mMaul}(R_{\mathsf{A}}, r'_0 \oplus r_2^{-1})$ is uniformly random over $\mathcal{R}_{cm}$. By the tightness of Π'_{Com}, for $\overline{C}'$ and R'_{A}, the opening message $\overline{x}' = \mathsf{OpenRerand}(\mathsf{xMaul}(\overline{x}, r'_0 \oplus r_2^{-1}), r'_1)$ is unique and uniformly random.

The strong sanitation of F_j is proved similarly. First, an input-trigger attack against C_j is infeasible: $u'' = (C'', c''_j, \overline{C}'')$ and $(x'', \overline{x}'')$ received by C_j are always unbiased due to rerandomization via $\bar{r}_1, \bar{r}'_1$ and $\bar{\gamma}_j$. By the key malleability of bKEM, $\widetilde{pk}'' = \mathsf{KeyMaul}(\widetilde{pk}', \bar{r}_0)$ is uniformly random. Therefore, $\widetilde{K} = \mathsf{bKEM.Decap}(\widetilde{sk}'', \tilde{c}) = \mathsf{bKEM.Decap}(\widetilde{sk}', \tilde{c}')$ is also uniformly random where $\widetilde{sk}''$ is the corresponding private key of $\widetilde{pk}''$. The underlying key of c'_j is mauled into K''_j by $\bar{r}_2$. Since KEM is ciphertext-malleable, K''_j has the same distribution as $\widehat{K}_j$. C_j gets $K'_i = \mathsf{KMaul}(K_i, R''_{\mathsf{A}} \oplus R_{\mathsf{B}})$, $K'_j = \mathsf{KMaul}(K''_j, R''_{\mathsf{A}} \oplus R_{\mathsf{B}})$, where $R''_{\mathsf{A}} \oplus R_{\mathsf{B}}$ is uniformly random. Therefore, K'_i and K'_j computed by C_j are unbiased. Second, v' output by F_j is unbiased: F_j sanitizes $v = (\tilde{c}, c_i, R_{\mathsf{B}})$ (received from C_j) into $v' = (\tilde{c}', c'_i, R'_{\mathsf{B}})$ by computing $\tilde{c}' \leftarrow \mathsf{CKeyMaul}(\tilde{c}, \bar{r}_0)$, $c'_i \leftarrow \mathsf{Rer}(\mathsf{CMaul}(c_i, \bar{r}_2^{-1}), \bar{\gamma}_i)$ and $R'_{\mathsf{B}} = R_{\mathsf{B}} \oplus \bar{r}'_0$. Since $\widetilde{K} = \mathsf{bKEM.Decap}(\widetilde{sk}'', \tilde{c}) = \mathsf{bKEM.Decap}(\widetilde{sk}', \tilde{c}')$ is uniformly random, $\tilde{c}'$ is uniformly random (for a fixed $\widetilde{sk}'$). By the perfect rerandomizability of KEM, the distributions of c'_i and $\hat{c}_i$ are identical. R'_{B} is uniformly random due to the use of $\bar{r}'_0$. □

Lemma 3. *The firewalls F_i and F_j are computationally transparent.*

Proof. We begin by analyzing the computational transparency of F_i. We show that for all environments $\mathcal{Z}$, it holds that $\text{EXEC}_{\mathsf{srAKE},\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathsf{srAKE}',\mathcal{A},\mathcal{Z}}$, where $\mathcal{A}$ is the dummy adversary, srAKE and srAKE′ run with dummy parties except for U, which is either $(\mathsf{C}_i, \mathsf{F}_i)$ or $(\mathsf{C}_i, \mathsf{ID})$. By the definition of computational transparency, C_i,F_i and ID are uncorrupted. By Lemma 2, messages output by F_i and $K'_i, K'_j, \widetilde{K}$ computed by C_i are always unbiased, i.e., the distributions of the output of C_i in the presence of F_i are identical to that of the output of C_i in the presence of ID. Therefore, we have $\text{EXEC}_{\mathsf{srAKE},\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathsf{srAKE}',\mathcal{A},\mathcal{Z}}$. The transparency of F_j can be proved in the same way. □

We now present the simulation of an execution of srAKE in the corruption case where both the core and RF of one party are malicious, while the core and RF of the other party remain honest.

Lemma 4. *For every adversary $\mathcal{A}$ corrupting one party maliciously in an execution of* srAKE *in the random oracle model, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, it holds that* $\text{EXEC}_{\mathsf{srAKE},\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathsf{Wrap}(\mathcal{F}_{\mathsf{AKE}}),\mathcal{S},\mathcal{Z}}$.

Proof sketch. We consider two sub-cases: either U or S is corrupt. If one of the parties is corrupt, $\mathcal{S}$ communicates with the corrupt party on behalf of the honest one. By the definition of NEWKEY interface of $\mathcal{F}_{\mathsf{AKE}}$, when the corrupt

party interacts with an honest party, the session key is uniformly chosen by $\mathcal{F}_{\mathsf{AKE}}$. Therefore, $\mathcal{S}$ should ensure consistency between the specific RO answer and session key. Let K_i^*, K_j^* and $\widetilde{K}^*$ be the input keys of U to H. Let K_i^{**}, K_j^{**} and $\widetilde{K}^{**}$ be the input keys of S to H.

In the case of U-corruption, $\mathcal{S}$ simulates the behaviors of S. $\mathcal{A}$ is able to query $H(sid\|ssid\|\mathsf{aux}\|K_i^*\|K_j^*\|\widetilde{K}^*)$ upon receiving v' from $\mathcal{S}$. $\mathcal{S}$ is required to detect this RO query and answers with the session key. Therefore, upon receiving $u' = (C', c_j', \overline{C}')$ from $\mathcal{A}$, $\mathcal{S}$ should extract $(\widetilde{pk}', R_{\mathsf{A}}')$ from $(C', \overline{C}')$ and subsequently compute K_i^*, K_j^* and $\widetilde{K}^*$. Then before receiving x' and $\overline{x}'$, $\mathcal{S}$ is able to identify the RO query $H(sid\|ssid\|\mathsf{aux}\|K_i^*\|K_j^*\|\widetilde{K}^*)$ that corresponds to the session key. The extraction operation requires Π_{Com} to be extractable.

In the case of S-corruption, $\mathcal{A}$ should not be able to make an RO query $H(sid\|ssid\|\mathsf{aux}\|K_i^{**}\|K_j^{**}\|\widetilde{K}^{**})$ before receiving the opening $(x', \overline{x}')$ from $\mathcal{S}$. Otherwise, $\mathcal{S}$ cannot keep a consistency between the session key output by $\mathcal{F}_{\mathsf{AKE}}$ and that computed by $\mathcal{A}$. This is guaranteed by the computationally hiding property of Π_{Com}, which ensures that $\mathcal{A}$ cannot obtain any information about $\widetilde{pk}'$ from C'. The detailed proof is shown in the full version. □

It remains to prove that, when each party has an honest core and an honest RF, we can construct a simulator $\mathcal{S}$ (communicating with $\mathsf{Wrap}(\mathcal{F}_{\mathsf{AKE}})$) to simulate the execution of srAKE, such that $\mathcal{Z}$ can distinguish between the real-world protocol execution and the ideal-world process with negligible probability.

Lemma 5. *For every adversary $\mathcal{A}$ that does not corrupt any RFs or cores in an execution of* srAKE *in the random oracle model, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, it holds that* $\text{EXEC}_{\mathsf{srAKE},\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\mathsf{Wrap}(\mathcal{F}_{\mathsf{AKE}}),\mathcal{S},\mathcal{Z}}$.

Proof sketch. The construction of $\mathcal{S}$ is shown in Fig. 10. In essence, the difference from $\mathcal{Z}$'s perspective in both executions lies in the computation of session keys: Parties in the real-world protocol execution compute the session key via an RO query. However, in the ideal world, $\mathcal{F}_{\mathsf{AKE}}$ would generate a uniformly random key without querying RO. We use a sequence of "hybrid games" to demonstrate the indistinguishability between the real world and the ideal world. The first game is the real-world execution where $\mathcal{Z}$ interacts with $\mathcal{A}$ and srAKE, and the last game is the ideal-world execution where $\mathcal{Z}$ interacts with $\mathcal{S}$ and $\mathsf{Wrap}(\mathcal{F}_{\mathsf{AKE}})$. We prove that the probability of $\mathcal{Z}$ outputting 1 in the first game is negligibly close to that in the last game. The detailed hybrid arguments are shown in the full version.

Simulator. Initially, $\mathcal{S}$ chooses public/private key pairs for both parties. $\mathcal{S}$ also runs KGen and KGen' to obtain the extraction keys ek and ek', respectively. Then $\mathcal{S}$ simulates C_i and F_i to generate $u' = (C', c_j', \overline{C}')$. When session $\langle ssid, \mathsf{S}, \mathsf{U}\rangle$ receives $u'' = (C'', c_j'', \overline{C}'')$, $\mathcal{S}$ extracts $(\widetilde{pk}'', R_{\mathsf{A}}'')$ from $(C'', \overline{C}'')$, simulates C_j and F_j to generate v' and computes $K_i^{**}, K_j^{**}, \widetilde{K}^{**}$. Then $\mathcal{S}$ sends v' to $\mathcal{A}$. When session $\langle ssid, \mathsf{U}, \mathsf{S}\rangle$ receives $v'' = (\widetilde{c}'', c_i'', R_{\mathsf{B}}'')$, $\mathcal{S}$ computes $K_i^*, K_j^*, \widetilde{K}^*$ and simulates C_i and F_i to generate $(x', \overline{x}')$. If $(K_i^*, K_j^*, \widetilde{K}^*) \neq (K_i^{**}, K_j^{**}, \widetilde{K}^{**})$,

i.e., $\mathcal{A}$ impersonates S, $\mathcal{S}$ sends an IMPERSONATE message to $\mathcal{F}_{\mathsf{AKE}}$. Moreover, if $u'' \neq u$ is generated adversarially (i.e., u'' is not produced in the manner of a firewall), which means $\mathcal{A}$ impersonates U, then $\mathcal{S}$ sends an IMPERSONATE message to $\mathcal{F}_{\mathsf{AKE}}$. $\mathcal{S}$ sends a NEWKEY message to $\mathcal{F}_{\mathsf{AKE}}$ and updates the session record if $\mathcal{F}_{\mathsf{AKE}}$ returns a session key. Then $\mathcal{S}$ sends $(x', \overline{x}')$ to $\mathcal{A}$. When $\langle ssid, \mathsf{S}, \mathsf{U}\rangle$ receives $(x'', \overline{x}'')$, if it is invalid, $\mathcal{S}$ drops the message, else $\mathcal{S}$ sends a NEWKEY message to $\mathcal{F}_{\mathsf{AKE}}$ and updates the record if $\mathcal{F}_{\mathsf{AKE}}$ returns a session key. When a party $\mathcal{P}$ is COMPROMISED, $\mathcal{S}$ sends a COMPROMISE message to $\mathcal{F}_{\mathsf{AKE}}$, updates the session record if $\mathcal{F}_{\mathsf{AKE}}$ returns a key, and reveals the private key of $\mathcal{P}$ to $\mathcal{A}$.

$\mathcal{S}$ answers RO queries $(sid\|ssid\|\mathsf{aux}\|K_1\|K_2\|K_3)$ by identifying those which correspond to potential session key computations of session $\langle ssid, \mathsf{U}, \mathsf{S}\rangle$ or $\langle ssid, \mathsf{S}, \mathsf{U}\rangle$. If there exists a matching record in list L_H, $\mathcal{S}$ returns the key in that record. Else, $\mathcal{S}$ picks $\kappa^* \leftarrow_{\$} \{0,1\}^w$. If there is a session record $\langle ssid, \mathsf{U}, \mathsf{S}, \kappa\rangle$, and it holds that $(K_1, K_2, K_3) = (K_i^*, K_j^*, \widetilde{K}^*)$ where K_i^*, K_j^* and $\widetilde{K}^*$ have been computed upon receiving v'', which implies that this RO query is a potential computation of the key for session $\langle ssid, \mathsf{U}, \mathsf{S}\rangle$, $\mathcal{S}$ programs the session key to be κ. Symmetrically, if there is a session record $\langle ssid, \mathsf{S}, \mathsf{U}, \kappa\rangle$, and it holds that $(K_1, K_2, K_3) = (K_i^{**}, K_j^{**}, \widetilde{K}^{**})$ where K_i^{**}, K_j^{**} and $\widetilde{K}^{**}$ have been computed upon receiving u'', which implies that this RO query is a potential computation of the key for session $\langle ssid, \mathsf{S}, \mathsf{U}\rangle$, $\mathcal{S}$ programs the session key to be κ. If $u'' \neq u'$ is generated adversarially, $\mathcal{A}$ compromises U and makes an RO query $(sid\|ssid\|\mathsf{aux}\|K_1\|K_2\|K_3)$ before sending $(x'', \overline{x}'')$ such that $(K_1, K_2, K_3) = (K_i^{**}, K_j^{**}, \widetilde{K}^{**})$, then $\mathcal{S}$ sends IMPERSONATE and NEWKEY messages to $\mathcal{F}_{\mathsf{AKE}}$ but delays instructing $\mathcal{F}_{\mathsf{AKE}}$ to output the session key to S. Upon receiving κ from $\mathcal{F}_{\mathsf{AKE}}$, $\mathcal{S}$ responds the RO query with κ. □

By combining Lemmas 1, 2, 3, 4 and 5 and the standard corruption translation table (Table 2), Theorem 1 follows immediately. □

7 Instantiations

Instantiation of bKEM. It is a straightforward construction of the DH-based KEM, denoted as DHK. The security of DHK is analyzed in the full version.

- $\mathsf{Setup}(1^\lambda)$: Randomly chooses a cyclic group $\mathbb{G}$ of order p with a generator g such that $2^\lambda \leq p < 2^{\lambda+1}$, sets $(\mathcal{K}, \mathcal{PK}, \mathcal{C}) := \{\mathbb{G}/1_{\mathbb{G}}\}^3, \mathcal{SK} := \mathbb{Z}_p^*$ and outputs $\mathsf{pp} = (\mathcal{K}, \mathcal{PK}, \mathcal{SK}, \mathcal{C}, p, g)$.
- $\mathsf{Gen}(\mathsf{pp})$: picks $k \leftarrow_{\$} \mathbb{Z}_p^*$ and outputs $(sk, pk) := (k, g^k)$.
- $\mathsf{Encap}(pk)$: picks $\beta \leftarrow_{\$} \mathbb{Z}_p^*$ and outputs $(c, K) := (g^\beta, pk^\beta)$.
- $\mathsf{Decap}(sk, c)$: outputs $K := c^{sk}$.
- $\mathsf{CKeyMaul}(c, r) := c^r$, $\mathsf{KeyMaul}(pk, r) := pk^r$.
- $\mathsf{Rg}(\mathsf{pp}) := r \leftarrow_{\$} \mathbb{Z}_p^*$, $\mathsf{Kg}(pk, r) := pk^r$, $\mathsf{Cg}(r) := g^r$.

Instantiation of KEM. KEM can be instantiated with particular KEM derived from the ElGamal cryptosystem, called $\mathsf{ElGamal\text{-}KEM}$. The security of $\mathsf{ElGamal\text{-}KEM}$ is analyzed in the full version.

Initially, $\mathcal{S}$ generates $(\rho, ek) \leftarrow\$ \mathsf{KGen}(1^\lambda)$, $(\rho', ek') \leftarrow\$ \mathsf{KGen}'(1^\lambda)$, $\mathsf{pp} \leftarrow \mathsf{bKEM.Setup}(1^\lambda), \mathsf{pp}' \leftarrow \mathsf{KEM.Setup}(1^\lambda)$. Then $\mathcal{S}$ generates $(pk_i, sk_i) \leftarrow\$ \mathsf{KEM.Gen}(\mathsf{pp}')$, $(pk_j, sk_j) \leftarrow\$ \mathsf{KEM.Gen}(\mathsf{pp}')$. $K_i^*, K_j^*, \widetilde{K}^*, K_i^{**}, K_j^{**}, \widetilde{K}^{**}$ and the list L_H are set to $\perp$.

On $(\textsc{UsrSession}, sid, ssid, \mathsf{U}, \mathsf{S})$ from $\mathcal{F}_{\mathsf{AKE}}$:

- Record $\langle ssid, \mathsf{U}, \mathsf{S}, \perp\rangle$. Generate the first message $u' = (C', c_j', \overline{C}')$ as U does in srAKE and send u' to $\mathcal{A}$ in the name of U.

On $(\textsc{SrvSession}, sid, ssid, \mathsf{U}, \mathsf{S})$ from $\mathcal{F}_{\mathsf{AKE}}$: Record $\langle ssid, \mathsf{S}, \mathsf{U}, \perp\rangle$.

On u'' from $\mathcal{A}$ to session $\langle ssid, \mathsf{S}, \mathsf{U}\rangle$: \\ Receiving the 1st message.

- Parse $u'' = (C'', c_j'', \overline{C}'')$. Generate $\widetilde{pk}'' \leftarrow \mathsf{Extract}(ek, C'')$, $R_{\mathsf{A}}'' \leftarrow \mathsf{Extract}(ek, \overline{C}'')$, and compute $v', \widetilde{K}^{**}, K_i^{**}$ and K_j^{**} as S does.
- Send v' to $\mathcal{A}$ in the name of S.

On v'' from $\mathcal{A}$ to session $\langle ssid, \mathsf{U}, \mathsf{S}\rangle$: \\ Receiving the 2nd message.

- Compute $(x', \overline{x}')$, $\widetilde{K}^*, K_i^*$ and K_j^* as U does in srAKE.
- If $(K_i^*, K_j^*, \widetilde{K}^*) \neq (K_i^{**}, K_j^{**}, \widetilde{K}^{**})$, send $(\textsc{Impersonate}, sid, ssid, \mathsf{S})$ to $\mathcal{F}_{\mathsf{AKE}}$. Furthermore, if $\widetilde{pk}'' \neq \widetilde{pk}' \vee K_j'' \neq K_j' \vee R_{\mathsf{A}}'' \neq R_{\mathsf{A}}'$, send $(\textsc{Impersonate}, sid, ssid, \mathsf{U})$ to $\mathcal{F}_{\mathsf{AKE}}$.
- Pick $\kappa^* \leftarrow\$ \{0,1\}^w$ and send $(\textsc{NewKey}, sid, ssid, \mathsf{U}, \kappa^*)$ to $\mathcal{F}_{\mathsf{AKE}}$. If $\mathcal{F}_{\mathsf{AKE}}$ returns κ, update record $\langle ssid, \mathsf{U}, \mathsf{S}, \perp\rangle$ to $\langle ssid, \mathsf{U}, \mathsf{S}, \kappa\rangle$.
- Send $(x', \overline{x}')$ to $\mathcal{A}$ in the name of U.

On $(x'', \overline{x}'')$ from $\mathcal{A}$ to session $\langle ssid, \mathsf{S}, \mathsf{U}\rangle$: \\ Receiving the 3rd message.

- If $\mathsf{Verif}(\rho, C'', x'') = \perp \vee \mathsf{Verif}(\rho', \overline{C}'', \overline{x}'') = \perp$, drop the message. \\ x'' or $\overline{x}''$ is invalid.
- If there is a record $\langle ssid, \mathsf{S}, \mathsf{U}, \kappa\rangle$, instruct $\mathcal{F}_{\mathsf{AKE}}$ to send $(sid, ssid, \kappa)$ to S.
- If $\exists\langle ssid, \mathsf{S}, \mathsf{U}, \perp\rangle$, pick $\kappa^* \leftarrow\$ \{0,1\}^w$ and send $(\textsc{NewKey}, sid, ssid, \mathsf{S}, \kappa^*)$ to $\mathcal{F}_{\mathsf{AKE}}$. If $\mathcal{F}_{\mathsf{AKE}}$ returns κ, update the record to $\langle ssid, \mathsf{S}, \mathsf{U}, \kappa\rangle$.

On $(\textsc{Compromise}, sid, \mathcal{P})$ from $\mathcal{A}$: \\ Compromise of a party $\mathcal{P}$.

- Send $(\textsc{Compromise}, sid, \mathcal{P})$ to $\mathcal{F}_{\mathsf{AKE}}$.
- If $\mathcal{F}_{\mathsf{AKE}}$ returns $(ssid, \mathcal{P}', \mathcal{P}, \kappa)$, update record $\langle ssid, \mathcal{P}', \mathcal{P}, \perp\rangle$ to $\langle ssid, \mathcal{P}', \mathcal{P}, \kappa\rangle$.
- Return the private key of $\mathcal{P}$ to $\mathcal{A}$.

On H query $(sid\|ssid\|\mathsf{aux}\|K_1\|K_2\|K_3)$: \\ Answering RO queries.

- If $\exists\ (sid\|ssid\|\mathsf{aux}\|K_1\|K_2\|K_3, \kappa^*) \in L_H$, return κ^*. Else pick $\kappa^* \leftarrow\$ \{0,1\}^w$.
- If $\exists\langle ssid, \mathsf{U}, \mathsf{S}, \kappa\rangle$, and $(K_1, K_2, K_3) = (K_i^*, K_j^*, \widetilde{K}^*)$, or $\exists\langle ssid, \mathsf{S}, \mathsf{U}, \kappa\rangle$ and $(K_1, K_2, K_3) = (K_i^{**}, K_j^{**}, \widetilde{K}^{**})$, set $\kappa^* \leftarrow \kappa$.
- If $\exists\langle ssid, \mathsf{S}, \mathsf{U}, \perp\rangle$, U is COMPROMISED and $(K_1, K_2, K_3) = (K_i^{**}, K_j^{**}, \widetilde{K}^{**}) \wedge (\widetilde{pk}'' \neq \widetilde{pk}' \vee K_j'' \neq K_j' \vee R_{\mathsf{A}}'' \neq R_{\mathsf{A}}')$, pick $\kappa' \leftarrow\$ \{0,1\}^w$, send $(\textsc{Impersonate}, sid, ssid, \mathsf{U})$ and $(\textsc{NewKey}, sid, ssid, \mathsf{S}, \kappa')$ to $\mathcal{F}_{\mathsf{AKE}}$. If $\mathcal{F}_{\mathsf{AKE}}$ returns κ, update the record to $\langle ssid, \mathsf{S}, \mathsf{U}, \kappa\rangle$ and set $\kappa^* \leftarrow \kappa$. \\ $\mathcal{A}$ impersonates U and queries $H(sid\|ssid\|\mathsf{aux}\|K_i^{**}\|K_j^{**}\|\widetilde{K}^{**})$ upon receiving v'.

Add $(sid\|ssid\|\mathsf{aux}\|K_1\|K_2\|K_3, \kappa^*)$ to L_H and output κ^*.

Fig. 10. Simulator $\mathcal{S}$ for srAKE when both parties are honest.

- $\mathsf{Setup}(1^\lambda)$: Randomly chooses a cyclic group $\mathbb{G}$ of order p s.t. $2^\lambda \le p < 2^{\lambda+1}$, sets $\mathcal{K}{:=}\mathbb{G}, \mathcal{SK}{:=}\mathbb{Z}_p, (\mathcal{PK}, \mathcal{C}){:=}\mathbb{G}^2$ and outputs $\mathsf{pp} = (\mathcal{K}, \mathcal{SK}, \mathcal{PK}, \mathcal{C}, p)$.
- $\mathsf{Gen}(\mathsf{pp})$: picks $k \leftarrow_\$ \mathbb{Z}_p, g \leftarrow_\$ \mathbb{G}$ and outputs $(sk, pk){:=}(k, (g, h = g^k))$.
- $\mathsf{Encap}(pk)$: parses $pk = (pk_1, pk_2)$, picks $\beta \leftarrow_\$ \mathbb{Z}_p, K \leftarrow_\$ \mathbb{G}$, computes $c \leftarrow (pk_1^\beta, pk_2^\beta \cdot K)$ and outputs (c, K).
- $\mathsf{Decap}(sk, c)$: parses $c = (c_1, c_2)$ and outputs $K{:=}c_2/c_1^{sk}$.

- $\mathsf{CMaul}(c, r)$: parses $c = (c_1, c_2)$, outputs $c' = (c_1, c_2 \cdot r)$.
- $\mathsf{KMaul}(K, r)$: outputs $K \cdot r$.
- $\mathsf{Rer}(c, r)$: parses $c = (c_1, c_2)$, outputs $c' = (c_1 \cdot g^r, c_2 \cdot h^r)$.

Instantiations of Π_{Com} and Π'_{Com}. Π_{Com} (see Table 3) is instantiated by ElGamal encryption. Π'_{Com} is the same as Π_{Com} except the following algorithms:

- $\mathsf{MaulCom}'(C, r)$: parse $C = (C_1, C_2)$, return $(C'_1, C'_2){:=}(C_1, C_2 \cdot r)$. ($\mathcal{R}'_{mcom} = \mathbb{G}$.)
- $\mathsf{mMaul}'(m, r)$: return $m'{:=}m \cdot r$. $\mathsf{xMaul}'(x, r)$: return x.

Table 3. Instantiation of Π_{Com}.

$\mathsf{KGen}(1^\lambda)$: Randomly choose a cyclic group $\mathbb{G}$ of order p with a generator g such that $2^\lambda \le p < 2^{\lambda+1}$. Set $\mathcal{M}_{com}{:=}\mathbb{G}, (\mathcal{R}_{com}, \mathcal{R}_{mcom}){:=}\mathbb{Z}_p^2, \mathcal{C}_{com}{:=}\mathbb{G}^2$. Then randomly pick $t \leftarrow_\$ \mathbb{Z}_p$, compute $h \leftarrow g^t$ as the CRS and output $\rho{:=}(\mathcal{M}_{com}, \mathcal{R}_{com}, \mathcal{R}_{mcom}, \mathcal{C}_{com}, g, h)$ and $ek{:=}t$.			
$\mathsf{Extract}(C, ek)$: parse $C = (C_1, C_2)$. output $m \leftarrow C_2/(C_1)^{ek}$.	$\mathsf{Com}(m, \rho; y)$: output $C{:=}(g^y, h^y \cdot m)$.	$\mathsf{Open}(\rho, C, y)$: output y.	$\mathsf{Verif}(\rho, C, y)$: parse $C = (C_1, C_2)$, if $C_1 = g^y$, output $m{:=}C_2/h^y$; otherwise, output $\perp$.
$\mathsf{Rerand}(C, r)$: parse $C = (C_1, C_2)$, compute $(C'_1, C'_2) \leftarrow (C_1 \cdot g^r, C_2 \cdot h^r)$.	$\mathsf{OpenRerand}(x, r)$: compute $x'{:=}(x + r)$ and return x'.	$\mathsf{MaulCom}(C, r)$: parse $C = (C_1, C_2)$, return $(C'_1, C'_2){:=}(C_1^r, C_2^r)$.	$\mathsf{mMaul}(m, r)$: output $m'{:=}m^r$. $\mathsf{xMaul}(x, r)$: return $x \cdot r$.

We defer the security analysis of Π_{Com} and Π'_{Com} to the full version. The instantiation of cores and RFs is also postponed to the full version.

Acknowledgments. We would like to thank anonymous reviewers of Eurocrypt 2025, Crypto 2025 and Asiacrypt 2025 for their helpful comments and suggestions. This work is supported in part by the National Natural Science Foundation of China (Grant No.62425205, No.62202485, No.62032005, No.62122092, and No.62372462) and the Young Elite Scientists Sponsorship Program by China Association for Science and Technology (No.YESS20230028).

References

1. Armour, M., Poettering, B.: Algorithm substitution attacks against receivers. Int. J. Inf. Sec. **21**(5), 1027–1050 (2022)
2. Arnold, P., Berndt, S., Müller-Quade, J., Ottenhues, A.: Protection against subversion corruptions via reverse firewalls in the plain universal composability framework. In: Fischlin, M., Moonsamy, V. (eds.) Applied Cryptography and Network Security, pp. 510–539. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-95764-2_20
3. Arnold, P., Berndt, S., Müller-Quade, J., Ottenhues, A.: Protection against subversion corruptions via reverse firewalls in the plain universal composability framework. Cryptology ePrint Archive, Paper 2023/1951 (2023). https://doi.org/10.1007/978-3-031-95764-2_20, https://eprint.iacr.org/2023/1951
4. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security, pp. 364–375. ACM Press, Denver, CO, USA (Oct 12–16) (2015). https://doi.org/10.1145/2810103.2813635
5. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 361–377. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005). https://doi.org/10.1007/11535218_22
6. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: 30th Annual ACM Symposium on Theory of Computing, pp. 419–428. ACM Press, Dallas, TX, USA (May 23–26, 1998). https://doi.org/10.1145/276698.276854
7. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security, pp. 1431–1440. ACM Press, Denver, CO, USA (Oct 12–16, 2015). https://doi.org/10.1145/2810103.2813681
8. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_1
9. Bemmann, P., Berndt, S., Chen, R.: Subversion-resilient signatures without random oracles. In: ACNS 24: 22nd International Conference on Applied Cryptography and Network Security, Part I, pp. 351–375. Lecture Notes in Computer Science, Springer, Cham, Switzerland (Jun 21–24, 2024). https://doi.org/10.1007/978-3-031-54770-6_14
10. Bemmann, P., Berndt, S., Diemert, D., Eisenbarth, T., Jager, T.: Subversion-resilient authenticated encryption without random oracles. In: Tibouchi, M., Wang, X. (eds.) ACNS 23: 21st International Conference on Applied Cryptography and Network Security, Part II. Lecture Notes in Computer Science, vol. 13906, pp. 460–483. Springer, Heidelberg, Germany, Kyoto, Japan (Jun 19–22, 2023). https://doi.org/10.1007/978-3-031-33491-7_17
11. Bemmann, P., Chen, R., Jager, T.: Subversion-resilient public key encryption with practical watchdogs. In: Garay, J. (ed.) PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 12710, pp. 627–658. Springer, Heidelberg, Germany, Virtual Event (May 10–13, 2021). https://doi.org/10.1007/978-3-030-75245-3_23

12. Berndt, S., Liskiewicz, M.: Algorithm substitution attacks from a steganographic perspective. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security, pp. 1649–1660. ACM Press, Dallas, TX, USA (Oct 31–Nov 2, 2017). https://doi.org/10.1145/3133956.3133981
13. - Berndt, S., Wichelmann, J., Pott, C., Traving, T.H., Eisenbarth, T.: ASAP: algorithm substitution attacks on cryptographic protocols. In: Suga, Y., Sakurai, K., Ding, X., Sako, K. (eds.) ASIACCS 22: 17th ACM Symposium on Information, Computer and Communications Security, pp. 712–726. ACM Press, Nagasaki, Japan (May 30–Jun 3, 2022). https://doi.org/10.1145/3488932.3517387
14. Bossuat, A., Bultel, X., Fouque, P.-A., Onete, C., van der Merwe, T.: Designing reverse firewalls for the real world. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) ESORICS 2020. LNCS, vol. 12308, pp. 193–213. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58951-6_10
15. Canetti, R.: Universally composable security. J. ACM **67**(5), 28:1–28:94 (2020)
16. Canetti, R., Jain, P., Swanberg, M., Varia, M.: Universally composable end-to-end secure messaging. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022, Part II. Lecture Notes in Computer Science, vol. 13508, pp. 3–33. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15979-4_1
17. Canetti, R., Krawczyk, H.: Universally Composable Notions of Key Exchange and Secure Channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_22
18. Chakraborty, S., Dziembowski, S., Nielsen, J.B.: Reverse firewalls for actively secure MPCs. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 732–762. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21) (2020). https://doi.org/10.1007/978-3-030-56880-1_26
19. Chakraborty, S., Ganesh, C., Pancholi, M., Sarkar, P.: Reverse firewalls for adaptively secure MPC without setup. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13091, pp. 335–364. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92075-3_12
20. Chakraborty, S., Ganesh, C., Sarkar, P.: Reverse firewalls for oblivious transfer extension and applications to zero-knowledge. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part I. Lecture Notes in Computer Science, vol. 14004, pp. 239–270. Springer, Heidelberg, Germany, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30545-0_9
21. Chakraborty, S., Magliocco, L., Magri, B., Venturi, D.: Key exchange in the post-snowden era: universally composable subversion-resilient PAKE. Cryptology ePrint Archive, Paper 2023/1827 (2023). https://eprint.iacr.org/2023/1827
22. Chakraborty, S., Magliocco, L., Magri, B., Venturi, D.: Key exchange in the post-snowden era: Universally composable subversion-resilient PAKE. In: Advances in Cryptology – ASIACRYPT 2024, Part V. Lecture Notes in Computer Science, vol. 15488, pp. 101–133. Springer, Singapore, Singapore (Dec 7–11, 2024). https://doi.org/10.1007/978-981-96-0935-2_4
23. Chakraborty, S., Magri, B., Nielsen, J.B., Venturi, D.: Universally composable subversion-resilient cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part I. Lecture Notes in Computer Science, vol. 13275, pp. 272–302. Springer, Heidelberg, Germany, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-06944-4_10

24. Chen, R., Huang, X., Yung, M.: Subvert KEM to break DEM: practical algorithm-substitution attacks on public-key encryption. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part II. Lecture Notes in Computer Science, vol. 12492, pp. 98–128. Springer, Heidelberg, Germany, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64834-3_4
25. Chen, R., Mu, Y., Yang, G., Susilo, W., Guo, F., Zhang, M.: Cryptographic reverse firewall via malleable smooth projective hash functions. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 844–876. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53887-6_31
26. Chow, S.S.M., Russell, A., Tang, Q., Yung, M., Zhao, Y., Zhou, H.-S.: Let a Non-barking Watchdog Bite: Cliptographic Signatures with an Offline Watchdog. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 221–251. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_8
27. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. J. Cryptol. **33**(4), 1914–1983 (2020). https://doi.org/10.1007/s00145-020-09360-1
28. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 767–797. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_25
29. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. Lecture Notes in Computer Science, vol. 9054, pp. 579–598. Springer, Heidelberg, Germany, Istanbul, Turkey (Mar 8–11, 2015). https://doi.org/10.1007/978-3-662-48116-5_28
30. Dodis, Y., Mironov, I., Stephens-Davidowitz, N.: Message transmission with reverse firewalls—secure communication on corrupted machines. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 341–372. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53018-4_13
31. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol. J. Cryptol. **34**(4), 37 (Oct 2021). https://doi.org/10.1007/s00145-021-09384-1
32. Faonio, A., Fiore, D., Herranz, J., Ràfols, C.: Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 159–190. Springer, Cham, Switzerland, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/978-3-030-34618-8_6
33. Fischlin, M., Mazaheri, S.: Self-guarding cryptographic protocols against algorithm substitution attacks. In: Chong, S., Delaune, S. (eds.) CSF 2018: IEEE 31st Computer Security Foundations Symposium, pp. 76–90. IEEE Computer Society Press, Oxford, UK (Jul 9–12, 2018). https://doi.org/10.1109/CSF.2018.00013
34. Galteland, H., Gjøsteen, K.: Subliminal channels in post-quantum digital signature schemes. Cryptology ePrint Archive, Report 2019/574 (2019). https://eprint.iacr.org/2019/574

35. Ganesh, C., Magri, B., Venturi, D.: Cryptographic reverse firewalls for interactive proof systems. In: Czumaj, A., Dawar, A., Merelli, E. (eds.) ICALP 2020: 47th International Colloquium on Automata, Languages and Programming. LIPIcs, vol. 168, pp. 55:1–55:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Saarbrücken, Germany (Jul 8–11, 2020). https://doi.org/10.4230/LIPIcs.ICALP.2020.55
36. Gellert, K., Gjøsteen, K., Jacobsen, H., Jager, T.: On optimal tightness for key exchange with full forward secrecy via key confirmation. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part IV. Lecture Notes in Computer Science, vol. 14084, pp. 297–329. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38551-3_10
37. Gu, Y., Jarecki, S., Krawczyk, H.: KHAPE: Asymmetric PAKE from key-hiding key exchange. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part IV. Lecture Notes in Computer Science, vol. 12828, pp. 701–730. Springer, Heidelberg, Germany, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84259-8_24
38. Gu, Y., Jarecki, S., Krawczyk, H.: KHAPE: Asymmetric PAKE from key-hiding key exchange. Cryptology ePrint Archive, Paper 2021/873 (2021). https://eprint.iacr.org/2021/873
39. Hofheinz, D., Müller-Quade, J., Steinwandt, R.: Initiator-resilient universally composable key exchange. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003: 8th European Symposium on Research in Computer Security. Lecture Notes in Computer Science, vol. 2808, pp. 61–84. Springer, Heidelberg, Germany, Gjøvik, Norway (Oct 13–15, 2003). https://doi.org/10.1007/978-3-540-39650-5_4
40. Hülsing, A., Ning, K.C., Schwabe, P., Weber, F., Zimmermann, P.R.: Post-quantum WireGuard. In: 2021 IEEE Symposium on Security and Privacy, pp. 304–321. IEEE Computer Society Press, San Francisco, CA, USA (May 24–27, 2021). https://doi.org/10.1109/SP40001.2021.00030
41. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 456–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_15
42. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_33
43. Liu, C., Chen, R., Wang, Y., Wang, Y.: Asymmetric subversion attacks on signature schemes. In: Susilo, W., Yang, G. (eds.) ACISP 18: 23rd Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 10946, pp. 376–395. Springer, Heidelberg, Germany, Wollongong, NSW, Australia (Jul 11–13, 2018). https://doi.org/10.1007/978-3-319-93638-3_22
44. Liu, J., Chen, R., Wang, Y., Tang, X., Su, J.: Subversion-resilient authenticated key exchange with reverse firewalls. In: Liu, J.K., Chen, L., Sun, S.F., Liu, X. (eds.) Provable and Practical Security, pp. 181–200. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-96-0957-4_10
45. Liu, J., Wang, Y., Chen, R., Tang, X., Su, J.: srCPace: universally composable PAKE with subversion-resilience. In: Lin, D., Wang, M., Yung, M. (eds.) Information Security and Cryptology, pp. 211–231. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-96-4731-6_11

46. Liu, J., Wang, Y., Tang, X., Chen, R., Huang, X., Su, J.: srTLS: secure TLS handshake on corrupted machines. IEEE Trans. Dependable Secur. Comput. **22**(3), 3137–3154 (2025). https://doi.org/10.1109/TDSC.2024.3524681
47. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 657–686. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_22
48. Perlroth, N., Larson, J., Shane, S.: Secret documents reveal nsa campaign against encryption (2013). https://archive.nytimes.com/www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html
49. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Cliptography: Clipping the Power of Kleptographic Attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_2
50. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic semantic security against a kleptographic adversary. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security, pp. 907–922. ACM Press, Dallas, TX, USA (Oct 31– Nov 2, 2017). https://doi.org/10.1145/3133956.3133993
51. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Correcting subverted random oracles. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 241–271. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_9
52. Santos, B.F.D., Gu, Y., Jarecki, S., Krawczyk, H.: Asymmetric PAKE with low computation and communication. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part II. Lecture Notes in Computer Science, vol. 13276, pp. 127–156. Springer, Heidelberg, Germany, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07085-3_5
53. Shoup, V.: A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112 (2001). https://eprint.iacr.org/2001/112
54. Steinfeld, R., Baek, J., Zheng, Y.: On the Necessity of Strong Assumptions for the Security of a Class of Asymmetric Encryption Schemes. In: Batten, L., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 241–256. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45450-0_20
55. Teseleanu, G.: Threshold kleptographic attacks on discrete logarithm based signatures. In: Lange, T., Dunkelman, O. (eds.) Progress in Cryptology - LATINCRYPT 2017: 5th International Conference on Cryptology and Information Security in Latin America. Lecture Notes in Computer Science, vol. 11368, pp. 401–414. Springer, Heidelberg, Germany, Havana, Cuba (Sep 20–22, 2019). https://doi.org/10.1007/978-3-030-25283-0_21
56. Young, A., Yung, M.: The dark side of "Black-Box" cryptography or: should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_8
57. Young, A., Yung, M.: Kleptography: using cryptography against cryptography. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 62–74. Springer, Heidelberg, Germany, Konstanz, Germany (May 11–15, 1997). https://doi.org/10.1007/3-540-69053-0_6