srTLS: Secure TLS Handshake on Corrupted Machines

Jiahao Liu, Yi Wang, Xincheng Tang, Rongmao Chen, Xinyi Huang, and Jinshu Su, *Senior Member, IEEE*

***Abstract*—TLS 1.3 is widely used to realize secure communication over the Internet. Existing security analyses of TLS 1.3 primarily focus on its handshake protocol which is indeed an authenticated key exchange (AKE) protocol, and implicitly neglect the so-called subversion attacks (e.g., breaking TLS via Dual EC) in the real world. Reverse firewall (RF) is a prevalent approach to defend against subversion attack. To the best of our knowledge, the only two subversion-resilient AKE protocols with RFs are proposed by Dodis et al. (CRYPTO'16) and Bossuat et al. (ESORICS'20). The security of both protocols is proved under game-based model which is insufficient for the concurrent execution of multiple TLS instances in practice. In this paper, we propose srTLS, a variant of the TLS 1.3 full one round-trip time (1-RTT) handshake protocol with RFs under the universally composable (UC) model. In particular, we first present the ideal functionality of unilateral AKE $\mathcal{F}_{\mathsf{uaKE}}$. Then, we use RFs with outer transparency to circumvent the difficulty in sanitizing the messages of handshake protocol, and prove that srTLS UC-realizes $\mathcal{F}_{\mathsf{uaKE}}$ in the presence of subversion attacks. Finally, we integrate srTLS and existing subversion-resilient AKE protocols into TLS 1.3. The evaluation result demonstrates that srTLS achieves at least a 44.86% efficiency improvement over other subversion-resilient AKE protocols.**

***Index Terms*—Subversion attacks, reverse firewalls, universal composability security, authenticated key exchange, Transport Layer Security (TLS).**

I. Introduction

TLS 1.3 [1] has been widely adopted to establish secure channels over the Internet. Roughly, TLS 1.3 is composed of two sub-protocols: the *handshake* protocol and the *record* protocol, where the handshake protocol negotiates session key for message encryption in the record protocol.

Received 1 July 2024; revised 26 October 2024; accepted 12 December 2024. Date of publication 2 January 2025; date of current version 15 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62202485, Grant 62032005, Grant 62122092, and Grant 62372462, in part by the Young Elite Scientists Sponsorship Program by China Association for Science and Technology under Grant YESS20230028, in part by the Science and Technology Research Plan Program by NUDT under Grant ZK22-03, and in part by the National Natural Science Foundation of China under Grant 62425205. *(Corresponding authors: Yi Wang; Rongmao Chen; Jinshu Su.)*

Jiahao Liu, Yi Wang, Xincheng Tang, and Rongmao Chen are with the College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: liujiahao14@nudt.edu.cn; wangyi14@nudt.edu.cn; tangxincheng19@nudt.edu.cn; chromao@nudt.edu.cn).

Xinyi Huang is with the College of Cyber Security, Jinan University, Guangzhou 510632, China (e-mail: xyhuang81@gmail.com).

Jinshu Su is with the Academy of Military Science, Beijing 100832, China (e-mail: sjs@nudt.edu.cn).

Digital Object Identifier 10.1109/TDSC.2024.3524681

Existing security analyses [2], [3], [4] of TLS 1.3 primarily considered the handshake protocol which is essentially an authenticated key exchange (AKE) protocol. Notably, these analyses implicitly assume that the implementation of protocols strictly conforms to the specification. However, this turns out not to be the case in the real world. In particular, a powerful adversary might be able to undermine the claimed security of the protocol by substituting its implementation or interfering with the choice of parameters, which is known as *subversion attacks*. The most notorious example is the Dual Elliptic Curve Deterministic Random Bit Generator (Dual EC) [5]. Once an adversary determines the parameters used in Dual EC and obtains some Dual EC output bits, it is possible to predict the subsequent output bits. Noting that Dual EC is used by many real-world TLS 1.2 implementations including RSA BSAFE, Windows SChannel and OpenSSL, Checkoway et al. [6] showed the subversion attack against Dual EC could be exploited to break the security of TLS 1.2.

To defend against subversion attacks, at Eurocrypt 2015, Mironov and Stephens-Davidowitz introduced the concept of cryptographic *reverse firewalls* (RFs) [7]. Essentially, an RF is an intermediary entity positioned between a party and the external network, and "sanitizes" messages sent by the party to thwart the leakage of sensitive information. Great efforts have been made to design RFs for various cryptographic schemes and protocols [7], [8], [9], [10], [11], [12], [13], [14]. Since the handshake protocol in TLS 1.3 can be viewed as an AKE protocol, we focus on the subversion-resilient AKE protocols with RFs. Indeed, building RFs for AKE protocols is a non-trivial task. In specific, existing AKE protocols commonly rely on signatures or message authentication codes (MACs) over the transcript to protect against man-in-the-middle attacks [9], which also prevents the RF from performing sanitation on the transcript of the protocol.

The first subversion-resilient AKE protocol with RF (hereafter referred to as DMS) was introduced by Dodis et al. [9]. Essentially, DMS is adapted from the signed Diffie-Hellman (DH) key exchange, and is RF friendly due to the use of bilinear pairings. However, the time cost of the pairing operation is much higher than the exponentiation operation over cyclic group, and integrating DMS into TLS 1.3 requires extensive modifications. Later, Bossuat et al. [15] proposed a secure TLS-like protocol with enhanced RFs in the presence of subverted client only. The handshake part of this protocol is a unilateral AKE (hereafter referred to as BBFOM) where only the server authenticates itself. Comparing to original RF, the enhanced RF is equipped with a public/private key pair so that it could derive the special

key negotiated by the client and RF in the handshake part, and use this key to sanitize the messages in the record part. Moreover, for both DMS and BBFOM, their security is proved under game-based models and might not hold when multiple protocol instances are executed concurrently.

In this work, we are motivated to design subversion-resilient handshake protocol of unilateral authentication with RFs for TLS 1.3 under the universally composable (UC) model, as unilateral authentication (i.e., server-only authentication) is a more common case in reality than mutual authentication [3]. The reasons for not considering the record protocol include 1) certain authenticated encryption with associated data (AEAD) schemes for the record protocol satisfying the integrity of ciphertexts [16], [17] exhibit inherent resistance to subversion attacks, and 2) integrating RF into AEAD scheme inevitably requires the modification of the TLS message format, as indicated in [15]. Besides, we only consider full one round-trip time (1-RTT) handshake mode with (elliptic curve) Diffie–Hellman ephemeral key exchange ((EC)DHE). It is possible that subversion attacks occur in the pre-shared key (PSK) mode [18]. Nevertheless, as is similar to [3], we focus on the "cryptographic core" of TLS 1.3 and do not consider advanced protocol characteristics (including the PSK mode). The analysis of the PSK handshake mode is beyond the scope of this work. Note that there are pre-shared keys input to parties in PSK handshake mode, which is not captured by existing ideal functionalities for key exchange. We leave the definition of such an ideal functionality and the design of subversion-resilient, UC-secure handshake protocol in this mode as future directions.

A. Our Contributions

In this work, we present srTLS, a subversion-resilient handshake protocol designed specifically for TLS 1.3. Our approach involves modifying the TLS 1.3 handshake protocol to accommodate RFs. Furthermore, we provide a formal proof of the UC security of srTLS against subversion attacks and demonstrate its superior efficiency through extensive experimental evaluation when compared to DMS and BBFOM. Specifically, our contributions are three-fold:

- *We present the ideal functionality $\mathcal{F}_{\mathsf{uaKE}}$ of unilateral AKE:* $\mathcal{F}_{\mathsf{uaKE}}$ is adapted from the functionality of AKE by Canetti et al. [19] and precisely captures the security properties of an ideal unilateral AKE protocol. $\mathcal{F}_{\mathsf{uaKE}}$ differs from the original AKE functionality [19] in three aspects. First, the adversary is allowed to impersonate any client to communicate with a server, as clients are unauthenticated. Second, it captures mutual key confirmation by adding a ready state, following the definition of password-based AKE functionality in [20]. Third, it implies the non-deniability of authentication by allowing the adversary access to the certification functionality as a global subroutine, as in [21], [22].
- *We propose a subversion-resilient handshake protocol* srTLS *with RFs of outer transparency for TLS 1.3 under the UC model:* Due to the circular invocations between components in TLS 1.3, it is challenging to prove the UC security of the full 1-RTT handshake protocol. So, we modify the original TLS 1.3 handshake protocol and obtain a new handshake protocol Π_{uaKE}. In specific, we first separate the handshake protocol from the record protocol by removing the encryption of handshake messages, then use the global certification functionality $\mathcal{G}_{\mathsf{cert}}^{\mathsf{pid}}$ [21] to abstract public key infrastructure (PKI) and the bulletin-board functionality $\mathcal{G}_{\mathsf{bb}}$ [22] to bind parties' identities and public keys, instead of using concrete signature schemes.
  As previously mentioned, the difficulty in building RFs for the AKE protocol results from the fact that the computation of signatures or MACs over the transcript prohibits RFs from modifying messages. To circumvent this issue, we turn to consider the RF with outer transparency [23] that is able to return "feedback messages" to its party to ensure the freshness of randomness. Specifically, upon receiving a message from a party P, P's RF re-randomizes the message and returns the randomness back to P. Then, P is forced to use a fresh randomness rather than a possibly biased one. It is worth noting that P is aware of the existence of RF, while other entities would not notice such RF, and this is why it is called an RF with outer transparency. After equipping Π_{uaKE} with RFs, we obtain the full handshake protocol srTLS, and prove its UC security in the presence of subversion attacks by following Anorld et al.'s [23] approach.
- *We evaluate the performance of* srTLS *and compare it with existing subversion-resilient AKE protocols [9], [15]:* We compare our scheme with DMS and BBFOM in terms of both theoretical analysis and implementation evaluation. Specifically, we analyze the communication cost between parties and the computational cost of the parties and RFs. Our scheme outperforms the other two schemes in theoretical efficiency, as our scheme does not require bilinear pairings or a second layer of encryption. In experiment evaluation, we make reasonable modifications to DMS and BBFOM to fit TLS 1.3. For DMS, the signature over bilinear pairings is replaced with signature over transcript as in TLS 1.3, which requires the RFs to return a "feedback message" to parties. The signature verification step of the RFs is also removed, since the signature in TLS 1.3 is encrypted and cannot be verified by RFs. While these modifications introduce an exponentiation operation instead of a bilinear pairing for parties, and remove the bilinear pairing for RFs, evaluation indicates that the modified DMS exhibits significantly higher efficiency compared to the original DMS. The experiment results demonstrate that when both sides employ RFs, the execution time of srTLS is at least 44.86% faster compared to existing AKE schemes adapted for TLS 1.3.

B. Related Work

Countermeasures against subversion attacks: In 1990 s, Young and Yung [24], [25] revealed the adversarial usage of "black-box" cryptography and formalized this attack as *kleptography*. After the Snowden revelation in 2013, Bellare et al. [16] investigated the subversion attack against symmetric-key encryption schemes where they redefined this attack as

algorithm-substitution attack (ASA). There have been many works studying ASA against various cryptographic primitives, showing the power of subversion attacks [8], [17], [26], [27], [28], [29].

The disclosure of subversion attacks has prompted substantial efforts to devise effective strategies for subversion-resilient cryptography [7], [8], [10], [11], [12], [13], [14], [30], [31]. Given that subversion attacks involve a distinct type of adversary compared to classical attacks, making additional reasonable assumptions becomes crucial; otherwise, achieving meaningful security guarantees appears improbable. The various assumptions explored in these works, encompassing both trusted components [7], [8], [9], [10], [11], [12], [13], [14] and architectural requirements [30], [31], [32] rely on different aspects and are proved useful in various scenarios.

As one of the most effective way to defend against subversion attacks, RFs have been applied in many cryptographic schemes, including but not limited to signatures [8], multiparty computation (MPC) protocols [10], [11], [12], [14] and message transmission protocols [9]. Moreover, many other tools have been developed to resist subversion attacks, such as the watchdogs [30] and self-guarding mechanism [33].

Recently, Fischlin [34] introduced the concept of "stealth key exchange", which is similar to subversion in key exchange. The concept involves utilizing the nonce in TLS 1.3 handshake to clandestinely transport an additional key. Fischlin presented a security model for the stealth key exchange and rigorously demonstrates the security of the stealth key generated from the modified TLS 1.3 in his model. We become aware that Chakraborty et al. [35] proposed the first UC-secure subversion-resilient PAKE protocol in their previous model [13]. They followed the PAKE construction of Canetti et al. [36] and constructed RFs for it.

Subversion-resilient UC model: Chakraborty et al. [13] initiated the study of subversion-resilient UC model. To achieve subversion resilience, a party is divided into two components: a core and an RF. The core executes the protocol's code, while any communication with other cores is routed through the RF. The core and the RF could be corrupted independently. In contrast to the RF, the core allows *specious corruption* which captures the subversion attack. Moreover, subversion attacks considered in this model include two types: input subversion and randomness subversion. In input subversion, the adversary modifies the core's input to trigger the output of sensitive information. In randomness subversion, the randomness used in the core is biased so that the adversary could retrieve sensitive information from the transcript.

Later, Arnold et al. [23] proposed a simpler model that captures subversion attacks and RFs under the plain UC model at the cost of only considering the randomness subversion. The advantage of Arnold et al.'s model over Chakraborty et al.'s model is that it supports the reuse of existing results, including the universal composition theorem. That is, their main theorem says if the RFs satisfy specific properties, then the new protocol equipped with these RFs UC-emulates the base protocol. To achieve subversion resilience for a UC-secure protocol, one only needs to construct RFs for the protocol that meet the specific requirements. Thus, Arnold et al. provided a modular way to build subversion-resilient UC-secure protocol which enables simple security analysis and efficient constructions.

TABLE I
COMPARISON OF AKE PROTOCOLS WITH RFS

Schemes	DMS [9]	BBFOM [15]	**Ours**
Auth.	mutual	unilateral	unilateral
Round	4 (+1)	3	3
UC-security	×	×	✓
Assumption	inverse CDH	DDH	dual-snPRF-ODH
Pairing-free	×	✓	✓
Compatibility with TLS	×	×	✓
Object of protection	Handshake only	Handshake + record layer	Handshake only
Considered ASAs	Client & server-side	Client-side only	Client & server-side

In this comparison, the term "Auth" denotes the authentication type of the protocol, including mutual authentication and unilateral authentication. The communication round of DMS is 5 when embedded into TLS due to a clientfinished message. Note that the "assumption" does not encompass assumptions on which the signature scheme is based.

Subversion-resilient AKE: Table I gives comparisons among protocols DMS [9], BBFOM [15] and our protocol with respect to security models, assumptions, efficiency, object of protections and considered types of subversions, etc. We give a detailed comparison of our protocol with DMS and BBFOM in the following respects.

- *Security model:* Our security model differentiates from that of both [9] and [15]. Both the DMS and BBFOM protocols were analyzed in the game-based security models, which could not guarantee the security when composed with or concurrently executed with other protocols. We instead analyze our protocol in the UC model and the result implies UC security with subversion resilience.
- *Compatibility/practicality:* In our context, a protocol's *compatibility* with the original protocol refers to the property that the transcript formats of both protocols are identical. We observe that when transferring DMS or BBFOM to the paradigm of TLS 1.3 protocol, the format of transcripts (including `ClientHello` and `ServerHello`) should be modified. Our protocol preserves the format of TLS messages transferred in the network. Our protocol's compatibility with the original TLS 1.3 protocol is due to the "feedback messages" of the RFs. It implies that to protect a party from subversion attacks, it is sufficient to adjust the implementation of the TLS 1.3 protocol on that specific party without the necessity of modifying its peer. This facilitates deployment in real-world scenarios.
- *Objects of protection:* Our work considers the subversion resilience only for the handshake protocol, which is also the case in DMS. However, BBFOM additionally constructed RFs for the record protocol at the cost of each RF holding a public/private key pair.
- *Type of subversion attacks considered:* Following Dodis et al. [9], our work considers possible ASAs against clients and servers, whereas BBFOM focused on protecting clients from subversion attacks. The core motivation behind BBFOM is the subverted client case [15].

It is important to note that in any protocol employing RFs, an RF is designed to provide subversion resilience for a single party only. Consequently, if the requirement is for both parties to exhibit subversion resilience, then RFs should be implemented on both sides.

Code: Our implementation code is available on Gitee: https://gitee.com/wx_974c3e6b28/tls_-against_-asa.git.

C. Organization

We begin by introducing the notations that will be employed throughout this paper, along with other necessary preliminaries. Section III formulates an ideal functionality $\mathcal{F}_{\mathsf{uaKE}}$ for unilaterally authenticated key exchange, presents a modified TLS 1.3 handshake protocol Π_{uaKE} and proves the UC-security of Π_{uaKE}. In Section IV, we revisit and extend Arnold et al.'s theorem to our setting. In Section V-A, we outline the construction of RFs for Π_{uaKE} and the resulting protocol srTLS. Subsequently, in Section V-B, we demonstrate that srTLS is UC-secure in the presence of subversion attacks. The evaluation results are presented in Section VI.

II. Preliminaries

A. Notations

Let λ denote the security parameter throughout this paper. Let $\mathsf{negl}(\lambda)$ denote a negligible function of λ. Let $x \leftarrow \mathcal{X}$ denotes choosing an element x from a set $\mathcal{X}$ uniformly and randomly. For two positive integers a and b (assuming that $a < b$), let $[a, b]$ represent a set of integer $\{a, a+1, \ldots, b\}$. "PPT" is short for "probabilistic polynomial-time". Let $\mathcal{P} \circ \mathcal{Q}$ be a composition of a party $\mathcal{P}$ and a party $\mathcal{Q}$.

B. Cryptographic Assumptions

Definition 1 (PRF security [37]): Let $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a psedurandom function (PRF) with key space $\mathcal{K}$, domain $\mathcal{X}$ and range $\mathcal{Y}$. Security for f is defined via two experiments $\mathsf{Exp}_b^{\mathsf{PRF}}(b \in \{0,1\})$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- $\mathcal{C}$ samples and publishes public parameters pp to $\mathcal{A}$, where pp is determined by the security parameter λ.
- If $b = 0$, $\mathcal{C}$ samples $k \leftarrow \mathcal{K}$ and sets $g(\cdot) := f(k, \cdot)$. Else $\mathcal{C}$ chooses a random function $g : \mathcal{X} \to \mathcal{Y}$.
- $\mathcal{A}$ has access to oracle g, i.e., $\mathcal{A}$ sends a query $x \in \mathcal{X}$, $\mathcal{C}$ returns $g(x)$.
- Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Let W_b denote the probability that $\mathcal{A}$ outputs 1in experiment $\mathsf{Exp}_b^{\mathsf{PRF}}$. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A},f}^{\mathsf{PRF}}(\lambda) := |W_0 - W_1|$. We say f is secure if for all PPT adversaries $\mathcal{A}$, it holds that $\mathsf{Adv}_{\mathcal{A},f}^{\mathsf{PRF}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Definition 2 (snPRF-ODH and dual-snPRF-ODH assumptions [2]): Let $\mathbb{G}$ be a cyclic group of prime order q with generator g. Let $f : \mathbb{G} \times \{0,1\}^* \to \{0,1\}^\lambda$ be a PRF where $\lambda \in \mathbb{N}$. The snPRF-ODH security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is defined as follows.

- $\mathcal{C}$ picks $b \leftarrow \{0,1\}$, $(u, v) \leftarrow \mathbb{Z}_q^2$ and sends $(\mathbb{G}, g, g^u, g^v)$ to $\mathcal{A}$. Then $\mathcal{A}$ returns a challenge label x^* to $\mathcal{C}$.
- $\mathcal{C}$ computes $y_0^* \leftarrow f(g^{uv}, x^*)$, $y_1^* \leftarrow \{0,1\}^\lambda$ and returns y_b^* to $\mathcal{A}$.
- $\mathcal{A}$ is given a PRF oracle, in which if $\mathcal{A}$ queries (S, x) such that $S \in \mathbb{G}$ and $(S, x) \neq (g^v, x^*)$, then $\mathcal{C}$ returns $y \leftarrow f(S^u, x)$.
- $\mathcal{A}$ stops and outputs a guess $b' \in \{0,1\}$.

The snPRF-ODH advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A},\mathbb{G},f}^{\mathsf{snPRF\text{-}ODH}}(\lambda) := 2 \cdot \Pr[b' = b] - 1$. The snPRF-ODH assumption holds if for all PPT adversaries $\mathcal{A}$, it holds that $\mathcal{A}_{\mathcal{A},\mathbb{G},f}^{\mathsf{snPRF\text{-}ODH}}(\lambda) \leq \mathsf{negl}(\lambda)$. The dual-snPRF-ODH assumption is the snPRF-ODH assumption for a function $f : \{0,1\}^* \times \mathbb{G} \to \{0,1\}^\lambda$ with swapped inputs, where the first input is a label and the second input is a key.

C. Message Authentication Codex (MAC)

Definition 3 (MAC scheme [38]): A MAC scheme $\Pi := (\mathsf{MGen}, \mathsf{Mac}, \mathsf{MVrfy})$ includes three algorithms:

- $\mathsf{MGen}(\lambda)$: takes a security parameter λ as input and outputs a key K with $|K| \geq \lambda$.
- $\mathsf{Mac}(K, m)$: takes a key K and a message $m \in \{0,1\}^*$ as input and outputs a tag τ.
- $\mathsf{MVrfy}(K, m, \tau)$: takes a key K, a message m and a tag τ as input and outputs a bit $b \in \{0,1\}$. Specifically, $b = 1$ if τ is a valid MAC; otherwise, $b = 0$.

Correctness: For every λ, $K \leftarrow \mathsf{MGen}(\lambda)$, and every $m \in \{0,1\}^*$, it holds that $\mathsf{MVrfy}(K, m, \mathsf{Mac}(K, m)) = 1$.

Definition 4 (EUF-CMA Security of MAC [38]): Let $\Pi = (\mathsf{MGen}, \mathsf{Mac}, \mathsf{MVrfy})$ be a MAC scheme. We define the following experiment $\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathsf{EUF\text{-}CMA}}$ for an adversary $\mathcal{A}$ attacking Π with security parameter λ:

- Generating $K \leftarrow \mathsf{MGen}(\lambda)$.
- Given an input λ and an oracle access to $\mathsf{Mac}(K, \cdot)$, $\mathcal{A}$ is required to output a message-tag pair (m, τ). Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ sends to $\mathsf{Mac}(K, \cdot)$.
- Finally, if $\mathsf{MVrfy}(K, m, \tau) = 1$ and $m \notin \mathcal{Q}$, then the experiment outputs 1 and $\mathcal{A}$ wins; otherwise, the experiment outputs 0.

We define the advantage of $\mathcal{A}$ in winning the above experiment as $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{EUF\text{-}CMA}}(\lambda) = \Pr(\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathsf{EUF\text{-}CMA}} \Rightarrow 1)$. A MAC is existentially unforgeable under an adaptive chosen-message attack (EUF-CMA) if for all PPT adversaries $\mathcal{A}$, it holds that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathsf{EUF\text{-}CMA}}(\lambda) \leq \mathsf{negl}(\lambda)$.

D. Key Derivation Functions

In TLS 1.3, the key derivation process relies on a Key Derivation Function (KDF), specifically the HMAC-based KDF (HKDF) [39]. The HKDF operates on the extract-then-expand paradigm to generate keys. At its core lies the HMAC [40], which utilizes a cryptographic hash function denoted as $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ and is keyed by some key $K \in \{0,1\}^\lambda$. The HMAC tag on some message m is computed as $\mathsf{HMAC.Mac}(K, m) := \mathsf{H}((K \oplus \mathsf{opad}) || \mathsf{H}((K \oplus \mathsf{ipad}) || m))$, where opad and ipad are two λ-bit padding values, composed of repeated bytes 0x5c and 0x36, respectively [2].

HKDF comprises two principal functions: the extraction function $\mathsf{HKDF.Extract}(XTS, SKM)$, where XTS represents the

extractor salt and SKM denotes the source key material, outputting a pseudo-random key PRK. The second function is the expansion function $\mathsf{HKDF.Expand}(PRK, CTXinfo, L)$, which accepts a PRK from HKDF.Extract as one input and context information $CTXinfo$ as another. HKDF.Expand produces pseudo-random key material KM of length L bits. In TLS 1.3, L is omitted except during traffic key derivation, where $L = \lambda$ is otherwise the default parameter. Both HKDF.Extract and HKDF.Expand are instantiated with HMAC, in which $\mathsf{HKDF.Extract}(XTS, SKM) := \mathsf{HMAC.Mac}(XTS, SKM)$ and HKDF.Expand iteratively invokes HMAC to produce pseudorandom output of the desired length [2].

E. Subversion Resilience in the UC Model

Overview of the UC framework: The UC model is proposed to capture universal composability of cryptographic protocols. "Composability" means that a protocol instance maintains its security when sequentially composed with other arbitrary protocols, and "universal" means that its security is guaranteed even when run concurrently with other protocol instances. UC security is a simulation-based security notion used to model the objective of a protocol, denoted as π, as an *ideal functionality*, referred to as $\mathcal{F}$. In this context, a protocol π is tasked with completing a specific objective. Therefore, its execution is compared to that of an "ideal protocol" which represents the ideal process for achieving the objective. A key component of this ideal protocol is $\mathcal{F}$, which can be conceptualized as a trusted third party that receives input from protocol participants and returns computational results according to the prescribed objective. If π successfully emulates the ideal protocol, then the security of π is deemed to be no weaker than that of the ideal protocol for the given objective.

The global subroutines: In the basic UC theorem, the protocol π under analysis must be *subroutine-respecting*, which restricts its use in numerous cryptographic protocols. Loosely defined, a subroutine-respecting protocol π is self-contained, with each instance of π not relying on any subroutines, such as an ideal functionality, that could be utilized by other protocol instances. Clearly, many real-world protocols, including TLS 1.3, do not meet this requirement. Specifically, all servers that support TLS 1.3 obtain certificates from a PKI, resulting in publicly verifiable signatures. Fortunately, by the UC with Global Subroutines (UCGS) theorem proposed by Badertscher et al. [21], we can prove the UC security of a protocol that utilizes global subroutines.

UC with subversion resilience: We will now provide a concise overview of the model presented by Arnold et al. [23], denoted as ABMO, which captures the subversion attacks and RFs within the plain UC model.

1) Capturing Subversion Attacks: In the real world, subversion attacks occur when an adversary replaces the code of an honest party with a subverted implementation. The adversary may embed some additional information, denoted as inf, such as a private key, into the party's code. The additional information is assumed to be uniformly sampled from a set I. Specifically,

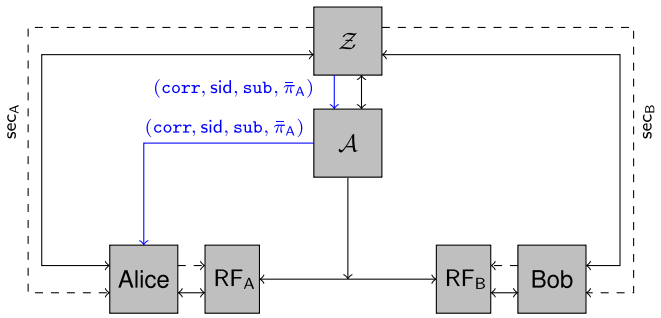


Fig. 1. RFs in the real world [23]. The dashed lines between the environment $\mathcal{Z}$ and parties represent the embedding of secrets by the environment into parties, which can potentially be exfiltrated through subversion. The dashed line between a party and its RF indicates the potential exfiltration of secrets. The blue lines depict the transmission of subversion messages. (For brevity, we only show the subversion against Alice as an example.).

in this scenario, an adversary $\mathcal{A}$ sends a subversion message $(\mathtt{corr}, \mathsf{sid}, \mathtt{sub}, \bar{\pi})$ to a party $\mathcal{P}$, where $\bar{\pi}$ is a subverted code, as shown in Fig. 1. Upon receiving this message, $\mathcal{P}$ replaces its original protocol implementation π with $\bar{\pi}$. The adversary can then extract the secret value either from the transcript or through interactions of the subverted party with malicious peers.

Specious corruptions: The subversion attacks discussed in [13], [23] are both described as "specious". Informally, a corruption of an adversary is called "specious" if the adversary substitutes the code $\pi_{\mathcal{P}}$ of a party $\mathcal{P}$ with another implementation $\pi'_{\mathcal{P}}$ that is indistinguishable from $\pi_{\mathcal{P}}$ through a black-box access to either $\pi_{\mathcal{P}}$ or $\pi'_{\mathcal{P}}$. The specious subversion is captured by the definition of *speciously subverted code*. Let π_i and $\bar{\pi}_i$ denote an honest implementation code and a subverted implementation code of a party $\mathcal{P}_i$, respectively. Let r denote the randomness used by the environment $\mathcal{Z}$. The running of the protocol with π_i or $\bar{\pi}_i$ is denoted by Π_i or Π'_i, respectively. The speciously subverted code is formally defined as follows.

Definition 5 (Speciously subverted code $\bar{\pi}$ [23]): $\bar{\pi}$ is a speciously subverted code, if and only if, for all PPT environments $\mathcal{Z}$, it holds that $1 - \Pr_{\mathsf{inf} \leftarrow I}[\mathrm{EXEC}_{\Pi_i, \mathcal{A}, \mathcal{Z}}(\lambda, r) \approx \mathrm{EXEC}_{\Pi'_i, \mathcal{A}, \mathcal{Z}}(\lambda, r)]$ is negligible, where $\mathcal{A}$ is the dummy adversary and, for sampled information inf, Π_i and Π'_i are protocols as described previously.

In the definition above, $\mathcal{Z}$ has no access to the additional information inf. In the ideal world, upon receiving a subversion message $(\mathtt{corr}, \mathsf{sid}, \mathtt{sub}, \bar{\pi})$ from the environment, the simulator only sends a simple subversion message to the ideal functionality. Consequently, in the ideal world, neither the ideal functionality nor any other entity obtains access to either the subverted code $\bar{\pi}$ or the secret value sec. This is because subversion attacks occur when implementations of protocols are subverted. However, since there is no protocol specification in the ideal world, even the ideal functionality does not hold any secret.

Generally, there are two types of subversions. The first type involves subverting the input to a party, where the subverted code replaces the input with a secret value that may be exposed to the adversary during protocol execution. The second type involves manipulating the randomness used in the protocol. For

instance, the subverted code may employ rejection sampling [41] (repeatedly selecting random values until an output that meets specific criteria is obtained) and subsequently exfiltrate secret through the output. In the ABMO model, only the second type of subversion is considered, which implies that replacing input values is always regarded as suspicious behavior.

2) RFs in the UC Model: Since the ABMO model solely focuses on subversions of randomness, it requires that the subverted code always produces valid outputs [23]. Additionally, in the ideal world, there is no need for the ideal functionality to introduce a new interface specifically for RFs to sanitize inputs intended for parties. As a result, RFs are not captured in the ideal world. In contrast, in the real-world setting of the ABMO model, RFs are treated as separate entities. To differentiate parties from RFs, we follow Arnold et al. [23] to rename both entities. Parties in a protocol are called *main parties*. A main party interacts with its associated RFs referred to as *sub-parties*, over a secure channel. The communication between these entities is illustrated in Fig. 1. There are two essential properties of RFs: anti-signalling and transparency.

Anti-signalling: The fundamental objective of RFs is to ensure that a subverted party behind an honest firewall remains indistinguishable from an honest party behind that firewall. This requirement is captured by the notion of *anti-signalling*.

Definition 6 (Anti-signalling [23]): Let $\mathcal{P}$ and $\mathcal{P}'$ be two parties. Let $\bar{\mathcal{P}}$ be a subverted version of $\mathcal{P}$, and $\mathsf{F}_{\mathcal{P}}$ be the RF of $\mathcal{P}$. Let $\widehat{\mathcal{P}}$ and $\widehat{\mathsf{F}}_{\mathcal{P}}$ be the incorruptible version of $\mathcal{P}$ and $\mathsf{F}_{\mathcal{P}}$, respectively. Let Π be the protocol run by $\bar{\mathcal{P}} \circ \mathsf{F}_{\mathcal{P}}$, and Π' be the same protocol with Π except that $\bar{\mathcal{P}} \circ \mathsf{F}_{\mathcal{P}}$ is replaced by $\widehat{\mathcal{P}} \circ \widehat{\mathsf{F}}_{\mathcal{P}}$. $\mathsf{F}_{\mathcal{P}}$ is called $\mathsf{anti-signalling}$ if, for all PPT environments $\mathcal{Z}$, it holds that $1 - \Pr_{\mathsf{inf} \leftarrow I}[\mathrm{EXEC}_{\Pi,\mathcal{A}(\mathsf{inf}),\mathcal{Z}}(\lambda, r) \approx \mathrm{EXEC}_{\Pi',\mathcal{A}(\mathsf{inf}),\mathcal{Z}}(\lambda, r)]$ is negligible, where $\mathcal{A}$ is a dummy adversary which has access to the additional information inf embedded in the subverted code $\bar{\pi}$.

Transparency: It is an optional yet desirable property that captures whether the RF is detectable by either party. Previous literature has defined two types of transparency. The first type requires that a party $\mathcal{Q}$ cannot distinguish whether another party $\mathcal{P}$, with an RF installed, is executed with the RF or not. This type of transparency is referred to as *outer transparency* by Arnold et al. [23]. Furthermore, there are RFs with another form of transparency, in which a party cannot distinguish whether it is equipped with an RF or not, referred to as *inner transparency* [23]. A firewall that has both inner transparency and outer transparency is said to provide *strong transparency*. A formal definition of transparency is shown as follows.

Definition 7 (Transparency [23]): Let $\mathcal{P}$ be a party and $\mathsf{F}_{\mathcal{P}}$ be its RF. Let $\widetilde{\mathcal{P}}$ and $\widetilde{\mathsf{F}}_{\mathcal{P}}$ be incorruptible versions of $\mathcal{P}$ and $\mathsf{F}_{\mathcal{P}}$, respectively. Let Π be a protocol run by $\widetilde{\mathcal{P}}$ and an incorruptible party $\mathcal{P}'$. Let Π' be a protocol identical to Π except that party $\widetilde{\mathcal{P}}$ is installed with $\widetilde{\mathsf{F}}_{\mathcal{P}}$. $\mathsf{F}_{\mathcal{P}}$ is said to provide $\mathsf{outer\ transparency}$ if, for all PPT environments $\mathcal{E}$, it holds that $\mathrm{EXEC}_{\mathcal{E},\Pi,\mathcal{A}}(\lambda, r) \approx \mathrm{EXEC}_{\mathcal{E},\Pi',\mathcal{A}}(\lambda, r)$, where $\mathcal{A}$ stands for the dummy adversary. Moreover, if the codes of $\mathcal{P}$ in Π and Π' are the same, then we say that $\mathsf{F}_{\mathcal{P}}$ has $\mathsf{strong\ transparency}$.

TABLE II
STANDARD CORRUPTION TRANSITION (ADAPTED FROM [23])

Party P	Firewall F	Composed corruption behavior
HONEST	HONEST	HONEST
HONEST	SEMIHONEST	HONEST
HONEST	MALICIOUS	**MALICIOUS**
SPECIOUS	HONEST	HONEST
SPECIOUS	SEMIHONEST	SEMIHONEST
SPECIOUS	MALICIOUS	MALICIOUS
MALICIOUS	-	MALICIOUS

"Specious" refers to specious subversions. "-" denotes any one of the following corruption types: honest, semi-honest and malicious corruptions.

Signature Generation
Upon receiving $(\textsc{Sign}, \mathsf{sid}, m)$ from party $\mathcal{P}$ with a party identifier pid:

- If this is the first request, then do:
  - If pid is not corrupted, then output $(\textsc{KeyGen})$ to the adversary. Upon receiving $(\textsc{Verif Key}, v)$ from the adversary, send $(\textsc{Register}, \mathsf{pid}, v)$ to $\mathcal{G}_{\mathsf{bb}}$.
  - Else send $(\textsc{Retrieve}, \mathsf{pid})$ to $\mathcal{G}_{\mathsf{bb}}$ and verify that $v \neq \perp$. If $v = \perp$, then ignore this request.
- Send $(\textsc{Sign}, \mathsf{sid}, m)$ to the adversary. Upon receiving $(\textsc{Signature}, \mathsf{sid}, m, \sigma)$ from the adversary, verify that no entry $(m, \sigma, 0)$ is recorded. If it is, then output $\perp$ to $\mathcal{P}$. Else, output $(\textsc{Signature}, \mathsf{sid}, m, \sigma)$ to $\mathcal{P}$, and record the entry $(m, \sigma, 1)$.

Signature Verification
Upon receiving a query $(\textsc{Verif}, \mathsf{sid}, m, \sigma)$ from party $\mathcal{P}$, check whether a pair (pid, v) is recorded. If no pair (pid, v) is recorded, then send $(\textsc{Retrieve}, \mathsf{pid})$ to $\mathcal{G}_{\mathsf{bb}}$ and obtain $(\textsc{Retrieve}, \mathsf{pid}, v)$.

- If $v = \perp$ then output $(\textsc{Verified}, \mathsf{sid}, m, 0)$.
- Else, record (pid, v) and send $(\textsc{Verif}, \mathsf{sid}, m, \sigma)$ to the adversary. Upon receiving $(\textsc{Verified}, \mathsf{sid}, m, \phi)$ from the adversary:
  - If (m, σ, b') is recorded, then set $f \leftarrow b'$.
  - Else if the signer is not corrupted, and no entry $(m, \sigma', 1)$ for any σ' is recorded, then set $f = 0$ and record the entry $(m, \sigma, 0)$.
  - Else set $f = \phi$, and record the entry (m, σ, f).

  Output $(\textsc{Verified}, \mathsf{sid}, m, f)$ to $\mathcal{P}$.

Fig. 2. The certification ideal functionality $\mathcal{G}_{\mathsf{cert}}^{\mathsf{pid}}$, which is a variant of that in [21]. It is parameterized by a party identity pid and only allows the party with pid to sign messages.

Corruption translation table: Let Π be a protocol without RFs and Π' be a protocol that adds RFs to Π. Since no RFs exist in Π, the adversary is allowed to corrupt parties and RFs in Π' but can only corrupt parties in Π. To prove that Π' UC-emulates Π, a corruption translation table is required to translate the corruption behaviors in Π' to those in Π. The corruption behaviors of the composition of a main party and its sub-party are shown in Table II. Parties in Π can be honest, semi-honest or maliciously corrupt, while parties in Π' can be honest, speciously or maliciously corrupt.

F. The Authentication Module

The classic authentication ideal functionality $\mathcal{F}_{\mathsf{auth}}$ is defined as a deniable task [42]: the receiver is unable to prove to any party that it receives a message from the sender. However, in a real-world setting utilizing PKI-based authentication, the receiver always obtains a transferable proof of the communication upon receiving a message. Therefore, it is infeasible for any protocol to realize the authentication functionality based solely on a plain PKI [43]. We begin by recalling the global certification ideal functionality $\mathcal{G}_{\mathsf{cert}}^{\mathsf{pid}}$ described in [21], [22], as shown in Fig. 2.

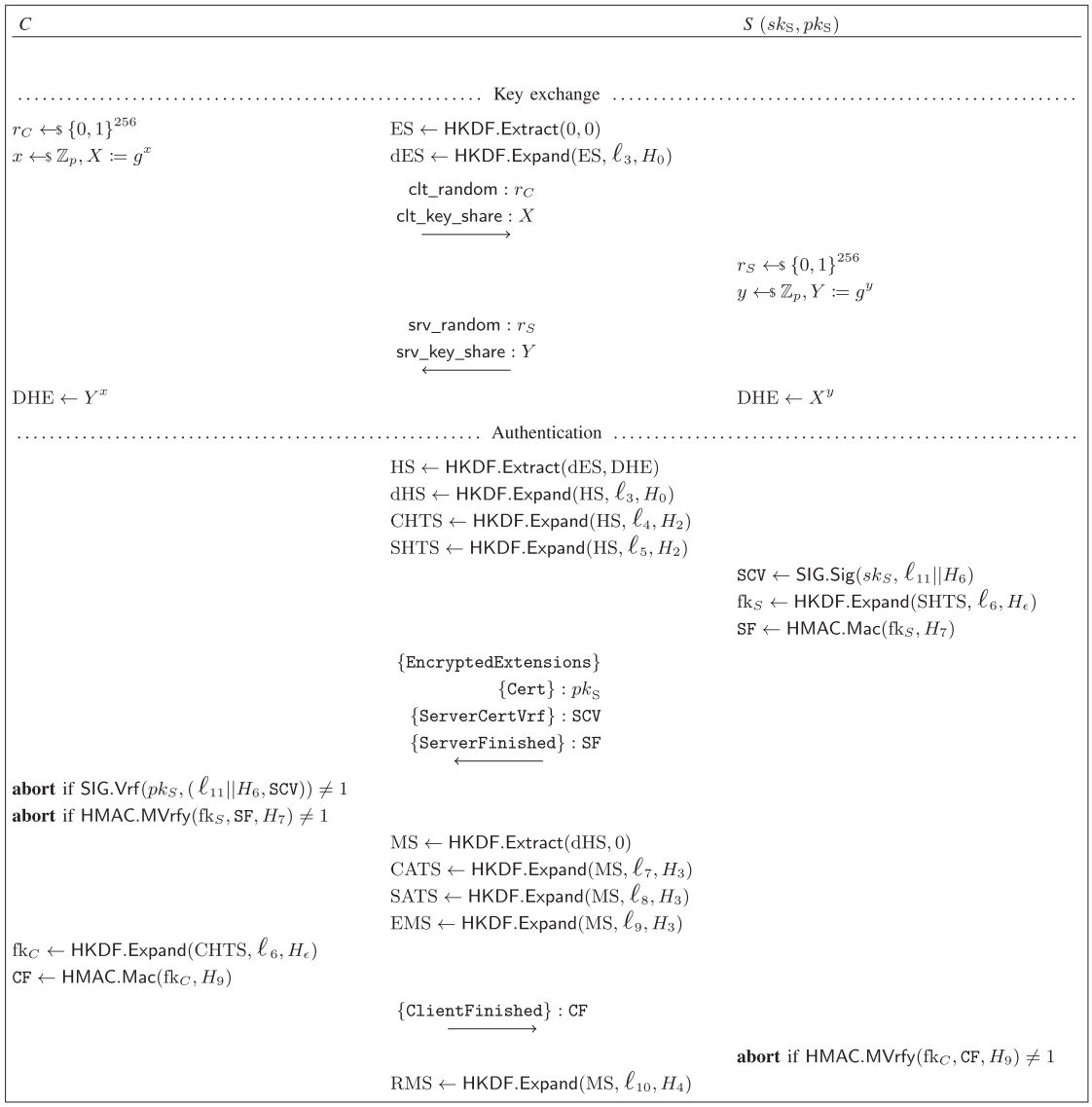


Fig. 3. The simplified TLS 1.3 handshake protocol with server-only authentication, where the `EncryptedExtensions` denotes extensions unnecessary for establishing the server handshake traffic key, and {`MSG`} refers to a message `MSG` encrypted with AEAD [2].

A key attribute of $\mathcal{G}_{\text{cert}}^{\text{pid}}$ is its global accessibility to any party, including the adversaries.

$\mathcal{G}_{\text{cert}}^{\text{pid}}$ is parameterized by a party identity pid. Different from $\mathcal{G}_{\text{cert}}^{\text{pid}}$ defined in [21], to model the certificate in the real world, it is required for $\mathcal{G}_{\text{cert}}^{\text{pid}}$ to use the *bulletin board functionality* $\mathcal{G}_{\text{bb}}$ [22], which serves as the certificate authority (CA). Specifically, $\mathcal{G}_{\text{bb}}$ works as follows [22]:

- *Report:* Upon receiving a message $(\text{REGISTER}, v)$ from party $\mathcal{P}$, send $(\text{REGISTERED}, \mathcal{P}, v)$ to the adversary. If the adversary responds with "OK" and this is the first request from $\mathcal{P}$, record the pair $(\mathcal{P}, v)$; otherwise, disregard the message.
- *Retrieve:* Upon receiving a message $(\text{RETRIEVE}, \mathcal{P})$ from a party $\mathcal{Q}$ (or the adversary $\mathcal{S}$), search for a record $(\mathcal{P}, v)$. If no such record exists, set $v = \perp$. Then, produce a public delayed output $(\text{RETRIEVE}, \mathcal{P}, v)$ to $\mathcal{Q}$ (or $\mathcal{S}$).

III. Modified TLS 1.3 Full 1-RTT Handshake

A. The TLS 1.3 Handshake Protocol

We show the TLS 1.3 handshake protocol in Fig. 3, which is an unilaterally (server-only) authenticated version of the TLS 1.3 handshake description in [2]. Let C and S denote the client and server, respectively. We consider a group $\mathbb{G}$ with prime order p and a generator g. The handshake is divided into two phases: the key exchange phase and the authentication phase. In the key exchange phase, both parties send their key shares and establish a DH secret. In the authentication phase, the server authenticates to the client and both send the FINISHED messages (i.e., `SF` and `CF`) as the last message. Traffic in the authentication phase are all encrypted by AEAD used in the record protocol. Table III shows abbreviations for messages and keys used in Fig. 3. The values of context and label inputs (H_*, ℓ_*) to HKDF.Expand

TABLE III
SHORTHANDS FOR TLS 1.3 DERIVED KEYS/VALUES (IN ALPHABETICAL ORDER) [2]

Derived key or value	Full name
CHTS/SHTS	Client/Server Handshake Traffic Secret
CATS/SATS	Client/Server Application Traffic Secret
dES/dHS	Derived Early/Handshake Secret
ES/HS/MS	Early/Handshake/Master Secret
EMS	Exporter Master Secret
$\mathsf{fk}_C/\mathsf{fk}_S$	Client/Server Finished Key
RMS	Resumption Master Secret
$\mathsf{tk}_{\mathsf{chs}}/\mathsf{tk}_{\mathsf{shs}}$	Client/Server Handshake Traffic Key
$\mathsf{tk}_{\mathsf{capp}}/\mathsf{tk}_{\mathsf{sapp}}$	Client/Server Application Traffic Key

TABLE IV
SECRET AND CORRESPONDING CONTEXT AND LABEL INPUTS, ALONG WITH THE COMPUTATION OF THE HANDSHAKE/APPLICATION TRAFFIC KEYS, WHERE L_k AND L_{iv} INDICATE THE KEY LENGTH AND IV LENGTH OF THE NEGOTIATED AEAD SCHEME, RESPECTIVELY [2].

Secret	Context input	Label input
dES	$H_0 = H(\text{“”})$	$\ell_3 =$ “derived”
CHTS	$H_2 = H(\text{ClientHello}\|\|\text{ServerHello})$	$\ell_4 =$ “c hs traffic”
SHTS	$H_2 = H(\text{ClientHello}\|\|\text{ServerHello})$	$\ell_5 =$ “s hs traffic”
fk_S	$H_\epsilon =$ “”	$\ell_6 =$ “finished”
dHS	$H_0 = H(\text{“”})$	$\ell_3 =$ “derived”
CATS	$H_3 = H(\text{ClientHello}\|\|\ldots\|\|\text{ServerFinished})$	$\ell_7 =$ “c ap traffic”
SATS	$H_3 = H(\text{ClientHello}\|\|\ldots\|\|\text{ServerFinished})$	$\ell_8 =$ “s ap traffic”
EMS	$H_3 = H(\text{ClientHello}\|\|\ldots\|\|\text{ServerFinished})$	$\ell_9 =$ “exp master”
RMS	$H_4 = H(\text{ClientHello}\|\|\ldots\|\|\text{ClientFinished})$	$\ell_{10} =$ “res master”
fk_C	$H_\epsilon =$ “”	$\ell_6 =$ “finished”
Auth. Value	**Context input**	**Context string**
SCV	$H_6 = H(\text{ClientHello}\|\|\ldots\|\|\text{ServerCert})$	ℓ_{11}=“TLS 1.3, server CertificateVerify”
SF	$H_7 = H(\text{ClientHello}\|\|\ldots\|\|\text{ServerCertVrf})$	
CF	$H_8 = H(\text{ClientHello}\|\|\ldots\|\|\text{ServerFinished})$	

$\mathsf{tk}_{\mathsf{chs}}/\mathsf{tk}_{\mathsf{shs}}/\mathsf{tk}_{\mathsf{capp}}/\mathsf{tk}_{\mathsf{sapp}} = (key, iv) = \mathsf{DeriveTK}(\text{CHTS/SHTS/CATS/SATS})$ where $\mathsf{DeriveTK}(\text{Secret}) = (\mathsf{HKDF.Expand}(\text{Secret}, \text{“key”}, H(\text{“ ”}), L_k), \mathsf{HKDF.Expand}(\text{Secret}, \text{“iv”}, H(\text{“ ”}), L_{iv}))$

and authentication functions, and the computation of traffic keys in Fig. 3 are shown in Table IV, where the gray text refers to the context and label inputs correspond to the generation of the handshake/application traffic secret.

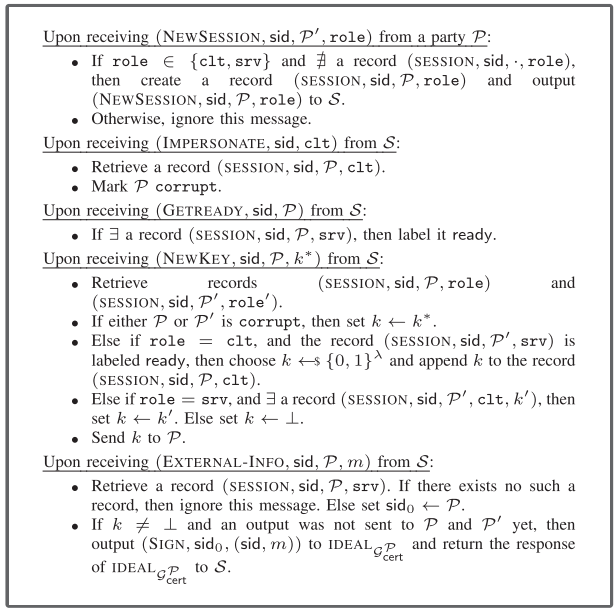
B. Ideal Functionality for Unilateral AKE

The unilaterally authenticated key exchange functionality $\mathcal{F}_{\mathsf{uaKE}}$ is shown in Fig. 4, where $\mathtt{role} \in \{\mathtt{clt}, \mathtt{srv}\}$. It generates session keys for both client and server, but with server-only authentication. Therefore, the adversary is allowed to impersonate a client to communicate with a server. Since we assume that servers authenticate via signing over transcripts (using public keys from PKI), this results in a transferable proof of authentication. To capture this transferability, we enable the ideal-world adversary to acquire signatures over arbitrarily selected messages within that session. At first sight, this approach may appear to contradict the authentication objective. However, it effectively preserves the fundamental aspects of authentication, as described by Canetti et al. [22].

Moreover, following the definition of PAKE functionality in [20], we capture mutual key confirmation in $\mathcal{F}_{\mathsf{uaKE}}$ by adding a ready state for the server, ensuring that (1) a client outputs a session key $k \neq \perp$ if and only if a server is in the ready state, and (2) a server outputs a session key $k \neq \perp$ if and only if a client has already output a key k', with k being set to k'.

C. Realizing $\mathcal{F}_{\mathsf{uaKE}}$

1) Unilaterally Authenticated Key Exchange: Fig. 5 depicts the Π_{uaKE} protocol analyzed in this paper, which is a variant



Upon receiving $(\text{NEWSESSION}, \mathsf{sid}, \mathcal{P}', \mathtt{role})$ from a party $\mathcal{P}$:
- If $\mathtt{role} \in \{\mathtt{clt}, \mathtt{srv}\}$ and $\nexists$ a record $(\text{SESSION}, \mathsf{sid}, \cdot, \mathtt{role})$, then create a record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{role})$ and output $(\text{NEWSESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{role})$ to $\mathcal{S}$.
- Otherwise, ignore this message.

Upon receiving $(\text{IMPERSONATE}, \mathsf{sid}, \mathtt{clt})$ from $\mathcal{S}$:
- Retrieve a record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{clt})$.
- Mark $\mathcal{P}$ $\mathtt{corrupt}$.

Upon receiving $(\text{GETREADY}, \mathsf{sid}, \mathcal{P})$ from $\mathcal{S}$:
- If $\exists$ a record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{srv})$, then label it ready.

Upon receiving $(\text{NEWKEY}, \mathsf{sid}, \mathcal{P}, k^*)$ from $\mathcal{S}$:
- Retrieve records $(\text{SESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{role})$ and $(\text{SESSION}, \mathsf{sid}, \mathcal{P}', \mathtt{role}')$.
- If either $\mathcal{P}$ or $\mathcal{P}'$ is $\mathtt{corrupt}$, then set $k \leftarrow k^*$.
- Else if $\mathtt{role} = \mathtt{clt}$, and the record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}', \mathtt{srv})$ is labeled ready, then choose $k \leftarrow\!\!\$\ \{0,1\}^\lambda$ and append k to the record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{clt})$.
- Else if $\mathtt{role} = \mathtt{srv}$, and $\exists$ a record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}', \mathtt{clt}, k')$, then set $k \leftarrow k'$. Else set $k \leftarrow \perp$.
- Send k to $\mathcal{P}$.

Upon receiving $(\text{EXTERNAL-INFO}, \mathsf{sid}, \mathcal{P}, m)$ from $\mathcal{S}$:
- Retrieve a record $(\text{SESSION}, \mathsf{sid}, \mathcal{P}, \mathtt{srv})$. If there exists no such a record, then ignore this message. Else set $\mathsf{sid}_0 \leftarrow \mathcal{P}$.
- If $k \neq \perp$ and an output was not sent to $\mathcal{P}$ and $\mathcal{P}'$ yet, then output $(\text{SIGN}, \mathsf{sid}_0, (\mathsf{sid}, m))$ to $\text{IDEAL}_{\mathcal{G}_{\mathsf{cert}}^{\mathcal{P}}}$ and return the response of $\text{IDEAL}_{\mathcal{G}_{\mathsf{cert}}^{\mathcal{P}}}$ to $\mathcal{S}$.

Fig. 4. The ideal functionality $\mathcal{F}_{\mathsf{uaKE}}$ for unilaterally authenticated key exchange. $\mathcal{F}_{\mathsf{uaKE}}$ initializes a value $k \leftarrow \perp$ at the beginning. The ideal protocol for $\mathcal{G}_{\mathsf{cert}}^{\mathcal{P}}$ is denoted as $\text{IDEAL}_{\mathcal{G}_{\mathsf{cert}}^{\mathcal{P}}}$.

of the handshake protocol in Fig. 3. Note that the description in Fig. 5 only shows the core cryptographic components of the handshake protocol and omits some specific computations and message fields which do not affect the security analysis. The main differences between Π_{uaKE} and the TLS 1.3 handshake (Fig. 3) are shown as follows.

Firstly, in Π_{uaKE}, both parties additionally have a session identifier sid as input. sid is transmitted along with the $\mathtt{ClientHello}/\mathtt{ServerHello}$ messages. In Π_{uaKE}, upon receiving as input $(\text{NEWSESSION}, \mathsf{sid}, S, \mathtt{clt})$, the client C generates clt_random and clt_key_share as in Fig. 3, and sends $(\mathsf{sid}, \mathsf{clt\_random}, \mathsf{clt\_key\_share})$ to a server S. Upon receiving as input $(\text{NEWSESSION}, \mathsf{sid}, \perp, \mathtt{srv})$, where $\perp$ refers to an anonymous client, the server S generates srv_random and srv_key_share as in Fig. 3, and sends $(\mathsf{sid}, \mathsf{srv\_random}, \mathsf{srv\_key\_share})$ to a client C.

Secondly, in Π_{uaKE}, we do not consider the encryption of handshake messages in the authentication phase. In the TLS 1.3 handshake, the handshake messages in the authentication phase are encrypted using the handshake traffic keys derived from the (EC)DHE shared secret. However, the encryption introduces a potential circular dependency between key exchange and authenticated encryption. This is due to the fact that the key used for encryption in the AEAD scheme is derived from the DH key shares, which are involved in the computation of the messages that AEAD encrypts. This circularity presents challenges in decomposing the TLS 1.3 full 1-RTT handshake, as both modules are interdependent. A similar circularity between authenticated encryption and key exchange module is also mentioned in the analysis of the Signal protocol [44]. In order to streamline the security analysis, we remove the encryption of handshake messages in the authentication phase. Fortunately, this modification does not compromise the secrecy of the session keys.

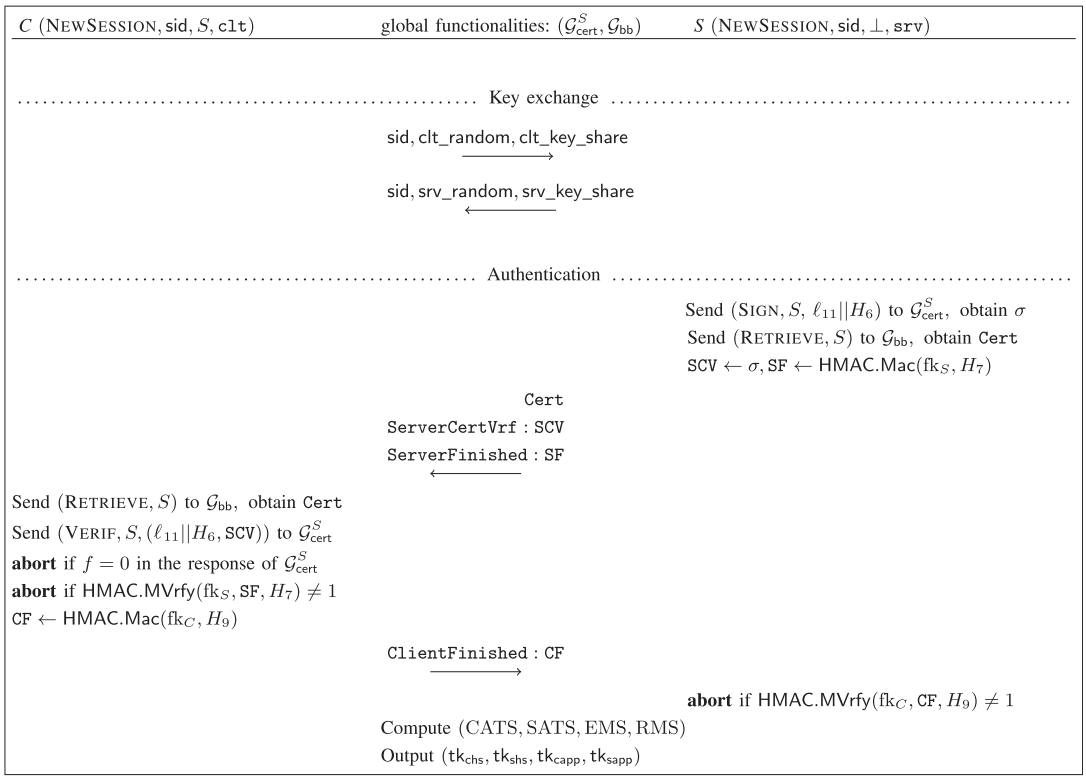


Fig. 5. Protocol Π_{uaKE} in the presence of global subroutines ($\mathcal{G}_{\mathsf{cert}}^{S}$, $\mathcal{G}_{\mathsf{bb}}$).

As demonstrated by Dowling et al. [2], the handshake traffic key provides security against passive adversaries, thereby enhancing the privacy of the handshake. However, the session keys would remain secret even if handshake messages are not encrypted. Note that the analysis of TLS 1.3 in [2], [3] also focused on the handshake protocol and ignored the record protocol.

Thirdly, PKI and server authentication are captured by global functionalities $\mathcal{G}_{\mathsf{bb}}$ [22] and $\mathcal{G}_{\mathsf{cert}}^{S}$ [21], respectively, instead of using a signature scheme. Formally, $\mathcal{G}_{\mathsf{bb}}$ and $\mathcal{G}_{\mathsf{cert}}^{S}$ are used in Π_{uaKE} as shared subroutines in the form of ideal protocols $\text{IDEAL}_{\mathcal{G}_{\mathsf{bb}}}$ and $\text{IDEAL}_{\mathcal{G}_{\mathsf{cert}}^{S}}$. For brevity, we refer to them directly as $\mathcal{G}_{\mathsf{bb}}$ and $\mathcal{G}_{\mathsf{cert}}^{S}$.

In the authentication phase, S sends (RETRIEVE, S) to $\mathcal{G}_{\mathsf{bb}}$ to obtain a certificate Cert, then sets $\mathsf{sid}_0 = S$ and sends (SIGN, $\mathsf{sid}_0, \ell_{11}||H_6||\mathsf{sid}$) to $\mathcal{G}_{\mathsf{cert}}^{S}$ for a certification of the transcript. Upon receiving (SIGNATURE, $\mathsf{sid}_0, \ell_{11}||H_6||\mathsf{sid}, \sigma$), S sets SCV := σ, computes SF and sends (Cert, SCV, SF) to the client.

On the client side, upon receiving (Cert, SCV, SF) from S, C sends (RETRIEVE, S) to $\mathcal{G}_{\mathsf{bb}}$ to acquire the certificate of S. Subsequently, C sets $\mathsf{sid}_0 = S$ and verifies SCV by sending (VERIF, $\mathsf{sid}_0, \ell_{11}||H_6||\mathsf{sid}$, SCV) to $\mathcal{G}_{\mathsf{cert}}^{S}$. Following this, C checks the correctness of SF.

If both SF and SCV are confirmed to be correct, C computes CF and sends it to S. Both parties can calculate the traffic keys ($\mathsf{tk}_{\mathsf{chs}}, \mathsf{tk}_{\mathsf{shs}}, \mathsf{tk}_{\mathsf{capp}}, \mathsf{tk}_{\mathsf{sapp}}$) as in Table IV. Since we focus on the TLS 1.3 full 1-RTT handshake mode, the 0-RTT (zero round-trip time) keys ETS (Early Traffic Secret) and EEMS (Early Exporter Master Secret) are not considered.

2) Security Proof: Now we prove the UC security of Π_{uaKE} under static corruptions, which means that the corruption of a party is determined at the start of the protocol execution.

Theorem 8: If the dual-snPRF-ODH assumption holds for HKDF.Extract and $\mathbb{G}$, HKDF.Extract and HKDF.Expand are secure PRFs, and HMAC is a secure MAC, then Π_{uaKE} (Fig. 5) UC-realizes $\mathcal{F}_{\mathsf{uaKE}}$ (Fig. 4) in the presence of global subroutines ($\mathcal{G}_{\mathsf{cert}}^{S}$, $\mathcal{G}_{\mathsf{bb}}$), where S is the party identifier of server.

Proof: We first construct a simulator $\mathcal{S}_{\mathsf{uaKE}}$ that interacts with the environment $\mathcal{Z}$ and $\mathcal{F}_{\mathsf{uaKE}}$, then show that $\mathcal{Z}$ has a negligible probability in distinguishing an interaction with Π_{uaKE} and $\mathcal{A}$ and an interaction with $\mathcal{F}_{\mathsf{uaKE}}$ and $\mathcal{S}_{\mathsf{uaKE}}$. $\mathcal{S}_{\mathsf{uaKE}}$ internally runs a protocol execution of Π_{uaKE} for $\mathcal{A}$, and acquires the required ServerCertVrf from the corresponding servers through $\mathcal{G}_{\mathsf{cert}}^{S}$. Specifically, $\mathcal{S}_{\mathsf{uaKE}}$ works as in Fig. 6.

We argue the indistinguishability through a series of hybrid games. Let $\mathsf{Real}_{\mathcal{Z}}(\Pi_{\mathsf{uaKE}}, \mathcal{A})$ denote the event that $\mathcal{Z}$ outputs 1in an interaction with $\mathcal{A}$ and an execution of Π_{uaKE}. Let $\mathsf{Ideal}_{\mathcal{Z}}(\mathcal{F}_{\mathsf{uaKE}}, \mathcal{S}_{\mathsf{uaKE}})$ denote the event that $\mathcal{Z}$ outputs 1in an interaction with $\mathcal{S}_{\mathsf{uaKE}}$ and $\mathcal{F}_{\mathsf{uaKE}}$. Let Hyb_i denote the event that $\mathcal{Z}$ outputs 1in Game G_i. In the cases of either party is corrupted, the simulation is trivial. Therefore, we only show detail of simulation in the case where both parties are honest.

G_0*:* The real-world execution of Π_{uaKE}. We have $\Pr[\mathsf{Hyb}_0] = \Pr[\mathsf{Real}_{\mathcal{Z}}(\Pi_{\mathsf{uaKE}}, \mathcal{A})]$.

G_1 *(introducing a simulator $\mathcal{S}$):* Π_{uaKE} is entirely run by a simulator $\mathcal{S}$ which includes $\mathcal{G}_{\mathsf{cert}}^{S}$ and $\mathcal{G}_{\mathsf{bb}}$. Since this change is only conceptual, we have $\Pr[\mathsf{Hyb}_1] = \Pr[\mathsf{Hyb}_0]$.

Upon receiving $(\text{NEWSESSION}, \mathsf{sid}, S, \mathtt{srv})$ from $\mathcal{F}_{\mathsf{uaKE}}$, record $(\mathsf{sid}, S, \mathtt{srv})$.
Upon receiving $(\text{NEWSESSION}, \mathsf{sid}, C, \mathtt{clt})$ from $\mathcal{F}_{\mathsf{uaKE}}$, send $(\mathsf{sid}, \mathsf{clt\_random}, \mathsf{clt\_key\_share})$ to $\mathcal{A}$, and record $(\mathsf{sid}, C, \mathtt{clt})$.
Upon receiving $(\mathsf{sid}, \mathsf{clt\_random}', \mathsf{clt\_key\_share}')$ from $\mathcal{A}$:

- pick $\mathsf{srv\_random} \leftarrow\!\$\ \{0,1\}^{256}$, $y \leftarrow\!\$\ \mathbb{Z}_p$ and $\mathsf{srv\_key\_share} \leftarrow g^y$.
- If $(\mathsf{clt\_random}', \mathsf{clt\_key\_share}') \neq (\mathsf{clt\_random}, \mathsf{clt\_key\_share})$, then send $(\text{IMPERSONATE}, \mathsf{sid}, \mathtt{clt})$ to $\mathcal{F}_{\mathsf{uaKE}}$, and set the flag $\mathsf{impersonate} = 1$.
- Send $(\text{RETRIEVE}, S)$ to $\mathcal{G}_{\mathsf{bb}}$ and obtains $\mathtt{Cert}$ from $\mathcal{G}_{\mathsf{bb}}$.
- Send $(\text{EXTERNAL-INFO}, \mathsf{sid}, S, \ell_{11}||H_6)$ to $\mathcal{F}_{\mathsf{uaKE}}$.
- Upon receiving $\mathtt{SCV}$ from $\mathcal{F}_{\mathsf{uaKE}}$, compute $\mathtt{SF} \leftarrow \mathsf{HMAC.Mac}(\mathsf{fk}_S, H_7)$ where $\mathsf{fk}_S \leftarrow\!\$\ \{0,1\}^L$ if $\mathsf{impersonate} = 1$, else fk_S is generated according to the protocol specification.
- Set $c := (\mathtt{Cert}, \mathtt{SCV}, \mathtt{SF})$, record and send $(\mathsf{sid}, \mathsf{srv\_random}, \mathsf{srv\_key\_share}, c)$ to $\mathcal{A}$.
- Send $(\text{GETREADY}, \mathsf{sid}, S)$ to $\mathcal{F}_{\mathsf{uaKE}}$.

Upon receiving $(\mathsf{sid}, \mathsf{srv\_random}, \mathsf{srv\_key\_share}, c)$ from $\mathcal{A}$,

- Parse c into $(\mathtt{Cert}, \mathtt{SCV}, \mathtt{SF}')$. If $\mathtt{SF}' \neq \mathtt{CF}$, then abort.
- Retrieve a record $(\mathsf{sid}, S, \mathtt{srv})$. Then set $\mathsf{sid}_0 \leftarrow S$ and $m' \leftarrow (\ell_{11}||H_6||\mathsf{sid})$, send $(\text{VERIF}, S, m', \mathtt{SCV})$ to $\mathcal{G}_{\mathsf{cert}}^{S}$.
- Upon receiving $(\text{VERIFIED}, \mathsf{sid}_0, m', f)$ from $\mathcal{G}_{\mathsf{cert}}^{S}$: If $f = 1$, S is honest and $(\mathsf{sid}, \mathsf{srv\_random}, \mathsf{srv\_key\_share}, c)$ was sent by S, then continue the subsequent computation; else abort.
- Compute $\mathtt{CF} \leftarrow \mathsf{HMAC.Mac}(\mathsf{fk}_C, H_9)$ where $\mathsf{fk}_C \leftarrow\!\$\ \{0,1\}^L$, then send $\mathtt{CF}$ to $\mathcal{A}$.
- Pick $(\mathsf{tk}_{\mathsf{chs}}, \mathsf{tk}_{\mathsf{shs}}, \mathsf{tk}_{\mathsf{capp}}, \mathsf{tk}_{\mathsf{sapp}})$ uniformly and randomly from the key space, then send $(\text{NEWKEY}, \mathsf{sid}, \mathtt{clt}, (\mathsf{tk}_{\mathsf{chs}}, \mathsf{tk}_{\mathsf{shs}}, \mathsf{tk}_{\mathsf{capp}}, \mathsf{tk}_{\mathsf{sapp}}))$ to $\mathcal{F}_{\mathsf{uaKE}}$.

Upon receiving $\mathtt{CF}'$ from $\mathcal{A}$,

- If $\mathtt{CF}' \neq \mathtt{CF}$, then abort.
- If $\mathsf{impersonate} = 1$, then compute $(\mathsf{tk}_{\mathsf{chs}}, \mathsf{tk}_{\mathsf{shs}}, \mathsf{tk}_{\mathsf{capp}}, \mathsf{tk}_{\mathsf{sapp}})$ according to the protocol specification. Else pick $(\mathsf{tk}_{\mathsf{chs}}, \mathsf{tk}_{\mathsf{shs}}, \mathsf{tk}_{\mathsf{capp}}, \mathsf{tk}_{\mathsf{sapp}})$ uniformly and randomly from the key space.
- Send $(\text{NEWKEY}, \mathsf{sid}, S, \mathtt{srv}, (\mathsf{tk}_{\mathsf{chs}}, \mathsf{tk}_{\mathsf{shs}}, \mathsf{tk}_{\mathsf{capp}}, \mathsf{tk}_{\mathsf{sapp}}))$ to $\mathcal{F}_{\mathsf{uaKE}}$.

Fig. 6. The construction of the simulator $\mathcal{S}_{\mathsf{uaKE}}$ for Π_{uaKE}.

G_2 *(introducing a functionality $\mathcal{F}$):* In this game, we separate a functionality $\mathcal{F}$ from $\mathcal{S}$, which has the same NEWSESSION, IMPERSONATE, GETREADY and NEWKEY interfaces with $\mathcal{F}_{\mathsf{uaKE}}$, but has no EXTERNAL-INFO interface. Moreover, upon receiving a NEWKEY query from $\mathcal{S}$, $\mathcal{F}$ sets the session key to be the value specified by $\mathcal{S}$. Since $\mathcal{S}$ can simulate all the messages correctly and determine the session keys, these changes make no difference to the view of $\mathcal{Z}$. Thus we have $\Pr[\mathsf{Hyb}_2] = \Pr[\mathsf{Hyb}_1]$.

G_3 *(replacing HS with a random string $\widetilde{\mathsf{HS}}$):*

Changes to $\mathcal{S}$: Upon receiving `ClientHello` generated by an honest client in session sid, or `ServerHello` generated by an honest server in session sid, the handshake secret HS is replaced by a uniformly random string $\widetilde{\mathsf{HS}} \leftarrow \{0,1\}^\lambda$.

If $\mathcal{Z}$ can distinguish between G_2 and G_3, we can build an adversary $\mathcal{B}_1$ against the dual-snPRF-ODH security of the HKDF.Extract function. $\mathcal{B}_1$ is constructed as follows. Firstly, $\mathcal{B}_1$ receives $(\mathbb{G}, g, W_1 = g^u, W_2 = g^v)$ from its challenger $\mathcal{C}$. Then $\mathcal{B}_1$ sets $X := W_1, Y := W_2$ where X and Y are the DH key shares of simulated C and S. $\mathcal{B}_1$ computes $\mathrm{dES} \leftarrow \mathsf{HKDF.Expand}(\mathrm{ES}, \ell_3, H_0)$ where $\mathrm{ES} \leftarrow \mathsf{HKDF.Extract}(0,0)$, and sends dES as the challenge label dES to $\mathcal{C}$. Upon receiving y^* from $\mathcal{C}$, $\mathcal{B}_1$ sets the handshake secret $\widetilde{\mathsf{HS}}$ to be y^*. The subsequent steps follow the specification of Π_{uaKE}. If $\mathcal{B}_1$ receives `ServerHello` message generated by $\mathcal{A}$, in which $Y' \neq Y$, it makes a query (Y', dES) to the PRF oracle and obtains the response as the handshake secret. There are two cases: If $b = 0$, then $\widetilde{\mathsf{HS}}$ is computed as $\widetilde{\mathsf{HS}} \leftarrow \mathsf{HKDF.Extract}(\mathrm{dES}, g^{uv})$, meaning that $\mathcal{B}_1$ perfectly simulates G_2. If $b = 1$, then $\widetilde{\mathsf{HS}}$ is a uniformly random value, making the simulation identical to G_3. Therefore, we have $|\Pr[\mathsf{Hyb}_3] - \Pr[\mathsf{Hyb}_2]| \leq \mathrm{Adv}^{\text{dual-snPRF-ODH}}_{\mathcal{B}_1, \mathbb{G}, \mathsf{HKDF.Extract}}(\lambda)$.

In the following games, the secrets and session keys of a party in a session are always set to match those of its peer if that session is not interfered with by the adversary and the peer has computed these values.

G_4 (replacing HKDF.Expand for $\widetilde{\mathsf{HS}}$):

Changes to $\mathcal{S}$: The PRF functions HKDF.Expand that take $\widetilde{\mathsf{HS}}$ as input are replaced. Specifically, instead of computing $\mathrm{CHTS}, \mathrm{SHTS}$ and dHS using HKDF.Expand, we pick $\mathrm{dHS}, \mathrm{CHTS}, \mathrm{SHTS} \leftarrow \{0,1\}^\lambda$ for randomly generated $\widetilde{\mathsf{HS}}$. Observe that since G_3, the handshake secret $\widetilde{\mathsf{HS}}$ has been uniformly sampled from $\{0,1\}^\lambda$ if the adversary is passive in the key exchange phase.

If $\mathcal{Z}$ can distinguish between G_4 and G_3, then we can build an adversary $\mathcal{B}_2$ to break the security of the HKDF.Expand function. Specifically, $\mathcal{B}_2$ simulates the protocol execution for $\mathcal{Z}$. When $\mathcal{B}_2$ is to compute dHS, it sends (ℓ_3, H_0) to its challenger $\mathcal{C}_2$. Upon receiving the response y from $\mathcal{C}_2$, $\mathcal{B}_2$ sets $\mathrm{dHS} \leftarrow y$. Similarly, when $\mathcal{B}_2$ is to compute CHTS (w.r.t. SHTS), $\mathcal{B}_2$ sends (ℓ_4, H_2) (w.r.t. (ℓ_5, H_2)) to $\mathcal{C}_2$ and sets CHTS (w.r.t. SHTS) to be the response. If $\mathcal{C}_2$ chooses $b = 0$, then $\mathcal{B}_2$ simulates G_3 perfectly. Else, $\mathrm{dHS}, \mathrm{CHTS}$ and SHTS are uniformly random, which implies that $\mathcal{Z}$ is in G_4. Therefore, we have $|\Pr[\mathsf{Hyb}_4] - \Pr[\mathsf{Hyb}_3]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_2, \mathsf{HKDF.Expand}}(\lambda)$.

G_5 (replacing HKDF.Expand for $\widetilde{\mathrm{CHTS}}$):

Changes to $\mathcal{S}$: The PRF functions HKDF.Expand that take $\widetilde{\mathrm{CHTS}}$ as input are replaced. Specifically, for randomly generated $\widetilde{\mathrm{CHTS}}$, instead of computing $\mathsf{tk}_{\mathsf{chs}}$ and fk_C using HKDF.Expand, we pick $\widetilde{\mathsf{tk}}_{\mathsf{chs}} \leftarrow \{0,1\}^L, \widetilde{\mathrm{fk}}_C \leftarrow \{0,1\}^\lambda$, where L represents the total length of the key and initialization vector in the agreed-upon AEAD scheme. If $\mathcal{Z}$ can distinguish G_5 from G_4, then we can build an adversary $\mathcal{B}_3$ to break the security of the HKDF.Expand function. The construction of $\mathcal{B}_3$ follows that of $\mathcal{B}_2$ in G_4. Thus we have $|\Pr[\mathsf{Hyb}_5] - \Pr[\mathsf{Hyb}_4]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_3, \mathsf{HKDF.Expand}}(\lambda)$.

G_6 (rejecting an HMAC forgery for $\widetilde{\mathrm{fk}}_C$): Upon receiving $\mathtt{CF}'$ from $\mathcal{A}$, $\mathcal{S}$ aborts if $\mathtt{CF}' \neq \mathtt{CF}$ is a valid MAC tag where $\mathtt{CF}$ is generated by $\mathcal{S}$. If $\mathcal{Z}$ can distinguish G_6 from G_5, then we can build a MAC forgery $\mathcal{B}_4$ using $\mathcal{Z}$ as a subroutine. Specifically, $\mathcal{B}_4$ simulates Π_{uaKE} as in G_6, except that when computing $\mathtt{CF}$, $\mathcal{B}$ queries $\mathsf{Mac}(K, \cdot)$ in the MAC security game and sets $\mathtt{CF}$ to be the output. If $\mathcal{A}$ sends $\mathcal{S}$ a message $\mathtt{CF}' \neq \mathtt{CF}$ and $\mathtt{CF}'$ is a valid MAC tag, then $\mathcal{B}_4$ returns $\mathtt{CF}'$ and the corresponding message m as a forgery, winning the MAC security game. Therefore, we have $|\Pr[\mathsf{Hyb}_6] - \Pr[\mathsf{Hyb}_5]| \leq \mathrm{Adv}^{\text{EUF-CMA}}_{\mathcal{B}_4, \mathsf{HMAC}}(\lambda)$.

G_7 (replacing HKDF.Expand for $\widetilde{\mathrm{SHTS}}$):

Changes to $\mathcal{S}$: The PRF functions HKDF.Expand that take $\widetilde{\mathrm{SHTS}}$ as input are replaced. Specifically, for randomly generated $\widetilde{\mathrm{SHTS}}$, instead of computing $\mathsf{tk}_{\mathsf{shs}}$ and fk_S using HKDF.Expand, we pick $\widetilde{\mathsf{tk}}_{\mathsf{shs}} \leftarrow \{0,1\}^L, \widetilde{\mathrm{fk}}_S \leftarrow \{0,1\}^\lambda$, where L represents the

total length of the key and initialization vector in the agreed-upon AEAD scheme. If $\mathcal{Z}$ can distinguish G_7 from G_6, then we can build an adversary $\mathcal{B}_5$ to break the security of the HKDF.Expand function. The construction of $\mathcal{B}_5$ follows that of $\mathcal{B}_2$ in G_4. Thus we have $|\Pr[\mathsf{Hyb}_7] - \Pr[\mathsf{Hyb}_6]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_5,\mathsf{HKDF.Expand}}(\lambda)$.

G_8 (rejecting an HMAC forgery for $\widetilde{\mathrm{fk}}_S$): Upon receiving $\mathtt{SF}'$ from $\mathcal{A}$, $\mathcal{S}$ aborts if $\mathtt{SF}' \neq \mathtt{SF}$ is a valid MAC tag where $\mathtt{SF}$ is generated by $\mathcal{S}$. If the abortion event happens, then we can build a MAC forgery $\mathcal{B}_6$ for HMAC using $\mathcal{Z}$ as a subroutine. Specifically, $\mathcal{B}_6$ simulates the protocol execution as in G_8, except that when $\mathcal{B}_6$ computes $\mathtt{SF}$, it makes a query to $\mathsf{Mac}(K,\cdot)$ in the MAC security game. If $\mathcal{A}$ sends a HMAC tag $\mathtt{SF}' \neq \mathtt{SF}$ that is valid, then $\mathcal{B}_6$ outputs $\mathtt{SF}'$ and the corresponding message and wins the MAC security game. Therefore, we have $|\Pr[\mathsf{Hyb}_8] - \Pr[\mathsf{Hyb}_7]| \leq \mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_6,\mathsf{HMAC}}(\lambda)$.

G_9 (replacing PRF in evaluating $\widetilde{\mathrm{dHS}}$): The PRF functions HKDF.Extract that take dHS as input are replaced. Specifically, for randomly generated $\widetilde{\mathrm{dHS}}$, instead of computing MS using HKDF.Extract, we pick $\widetilde{\mathrm{MS}} \leftarrow \{0,1\}^\lambda$. If $\mathcal{Z}$ can distinguish between G_9 and G_8, then we can build an adversary $\mathcal{B}_7$ to break the security of HKDF.Extract. This argument is analogous to that in G_4. Therefore, it holds that $|\Pr[\mathsf{Hyb}_9] - \Pr[\mathsf{Hyb}_8]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_7,\mathsf{HKDF.Extract}}(\lambda)$.

G_{10} (replacing PRF in evaluating $\widetilde{\mathrm{MS}}$): In this game, keys in all stages are uniformly random. The PRF functions HKDF.Expand that take $\widetilde{\mathrm{MS}}$ as input are replaced. Specifically, for randomly generated $\widetilde{\mathrm{MS}}$, instead of computing $\mathrm{CATS}, \mathrm{SATS}, \mathrm{EMS}, \mathrm{RMS}$ using HKDF.Expand, we pick $\mathrm{CATS}, \mathrm{SATS}, \mathrm{EMS}, \mathrm{RMS} \leftarrow \{0,1\}^\lambda$. If $\mathcal{Z}$ can distinguish the difference between G_{10} and G_9, then we can build an adversary $\mathcal{B}_8$ to break the security of HKDF.Expand. Following the same argument in G_4, we have $|\Pr[\mathsf{Hyb}_{10}] - \Pr[\mathsf{Hyb}_9]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_8,\mathsf{HKDF.Expand}}(\lambda)$. Observe that in G_{10}, all keys derived during the handshake protocol have been replaced by uniformly random values.

G_{11} (replacing HKDF.Expand for $\widetilde{\mathrm{CATS}}$):

Changes to $\mathcal{S}$: The PRF functions HKDF.Expand that take $\widetilde{\mathrm{CATS}}$ as input are replaced. Specifically, for randomly generated $\widetilde{\mathrm{CATS}}$, instead of computing $\mathsf{tk}_{\mathsf{capp}}$ using HKDF.Expand, we pick $\widetilde{\mathsf{tk}}_{\mathsf{capp}} \leftarrow \{0,1\}^L$, where L represents the total length of the key and initialization vector in the agreed-upon AEAD scheme. If $\mathcal{Z}$ can distinguish G_{11} from G_{10}, then we can build an adversary $\mathcal{B}_9$ to break the security of the HKDF.Expand function. The construction of $\mathcal{B}_9$ follows that of $\mathcal{B}_2$ in G_4. Thus we have $|\Pr[\mathsf{Hyb}_{11}] - \Pr[\mathsf{Hyb}_{10}]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_9,\mathsf{HKDF.Expand}}(\lambda)$.

G_{12} (replacing HKDF.Expand for $\widetilde{\mathrm{SATS}}$):

Changes to $\mathcal{S}$: The PRF functions HKDF.Expand that take $\widetilde{\mathrm{SATS}}$ as input are replaced. Specifically, for randomly generated $\widetilde{\mathrm{SATS}}$, instead of computing $\mathsf{tk}_{\mathsf{sapp}}$ using HKDF.Expand, we pick $\widetilde{\mathsf{tk}}_{\mathsf{sapp}} \leftarrow \{0,1\}^L$, where L represents the total length of the key and initialization vector in the agreed-upon AEAD scheme. If $\mathcal{Z}$ can distinguish G_{12} from G_{11}, then we can build an adversary $\mathcal{B}_{10}$ to break the security of the HKDF.Expand function. The construction of $\mathcal{B}_{10}$ follows that of $\mathcal{B}_2$ in G_4. Thus we have $|\Pr[\mathsf{Hyb}_{12}] - \Pr[\mathsf{Hyb}_{11}]| \leq \mathrm{Adv}^{\mathsf{PRF}}_{\mathcal{B}_{10},\mathsf{HKDF.Expand}}(\lambda)$.

G_{13} (removing $\mathcal{G}^S_{\mathsf{cert}}$ from $\mathcal{S}$):

Changes to $\mathcal{S}$: $\mathcal{G}^S_{\mathsf{cert}}$ runs independently from $\mathcal{S}$. When $\mathcal{S}$ needs to generate a signature on behalf of a server S, $\mathcal{S}$ sends $(\text{EXTERNAL-INFO}, \mathsf{sid}, S, m)$ to $\mathcal{F}$. The response of $\mathcal{F}$ is the signature over $m||\mathsf{sid}$.

Changes to $\mathcal{F}$: Add the EXTERNAL-INFO interface to $\mathcal{F}$ as in $\mathcal{F}_{\mathsf{uaKE}}$. Upon receiving $(\text{NEWKEY}, \mathsf{sid}, \mathcal{P}, k^*)$ from $\mathcal{S}$, do the same as in $\mathcal{F}_{\mathsf{uaKE}}$.

These changes to $\mathcal{S}$ and $\mathcal{F}$ are only conceptual and do not change the view of $\mathcal{Z}$. It holds that $\Pr[\mathsf{Hyb}_{13}] = \Pr[\mathsf{Hyb}_{12}]$. Now the behavior of $\mathcal{F}$ (resp. $\mathcal{S}$) is identical to that of $\mathcal{F}_{\mathsf{uaKE}}$ (resp. $\mathcal{S}_{\mathsf{uaKE}}$). Therefore, the view of $\mathcal{Z}$ in G_{13} is the same as that in the ideal world, i.e. $\Pr[\mathsf{Hyb}_{13}] = \Pr[\mathsf{Ideal}_{\mathcal{Z}}(\mathcal{F}_{\mathsf{uaKE}}, \mathcal{S}_{\mathsf{uaKE}})]$.

Thus, the advantage of $\mathcal{Z}$ in distinguishing the real-world execution from the ideal process is $|\Pr[\mathsf{Ideal}_{\mathcal{Z}}(\mathcal{F}_{\mathsf{uaKE}}, \mathcal{S}_{\mathsf{uaKE}})] - \Pr[\mathsf{Real}_{\mathcal{Z}}(\Pi_{\mathsf{uaKE}}, \mathcal{A})]| \leq \mathsf{negl}(\lambda)$. □

IV. Revisiting Arnold's Theorem

In this paper, we follow a modified version of the ABMO model that incorporates unauthenticated channels, global subroutines and RFs with only outer transparency, which means that a "feedback channel", through which an RF of a core can explicitly send extra messages to that core, is allowed. These changes are necessitated by the specific features of TLS 1.3, detailed in Section V-A.

Moreover, following the ABMO model, we consider only randomness subversions. It is practical for TLS 1.3 since a party's output value (i.e., session key) exclusively depends on the randomness choices of both parties rather than their inputs as in other MPC protocols. Therefore, we treat input substitutions as suspicious and do not capture RFs in the ideal world. We can utilize a standard ideal functionality and quantify over all PPT environments accordingly.

To construct a protocol Π' that UC-realizes a functionality $\mathcal{F}$ under specious subversions, we do as follows [23].

- Show that there is a protocol Π that UC-realizes $\mathcal{F}$.
- The corruption model used in the above UC analysis is extended to capture specious subversions. On the other hand, RFs are added to main parties in Π. The resulting protocol is denoted as Π'.
- Prove that Π' UC-emulates Π according to the corruption translations in Table II.
- It follows that Π' UC-realizes $\mathcal{F}$ by the transitivity of UC-emulation [45].

Arnold et al. [23] established that if the RFs exhibit strong transparency and anti-signalling properties, then the protocol equipped with RFs UC-emulates the original protocol in an authenticated channel setting against subversion attacks (as stated in Theorem 2 of [23], hereafter referred to as *UC-emulation theorem under specious subversions*). However, in the case of TLS 1.3, there are three differences which prevent us from applying this theorem directly:

- Firstly, strong transparency for RFs is hard to achieve. It seems necessary for a party's RF to provide feedback messages, such as the randomness used for re-randomization, to the party, consequently making the party aware of the existence of an RF and breaking the strong transparency.

- Secondly, TLS 1.3 works over an unauthenticated channel, which makes it necessary to re-prove the UC-emulation theorem under specious subversions in the unauthenticated setting.
- Finally, there are global subroutines in TLS 1.3, including the certification functionality and bulletin-board certificate authority functionality. Therefore, we need to show that the UC-emulation theorem under specious subversions holds in the presence of some global subroutines.

We demonstrate that the UC-emulation theorem under specious subversions holds despite the above differences.

Theorem 9: Let Π denote a protocol in the presence of global subroutines γ, with main parties P_i under party-wise malicious corruptions. Let Π' denote a protocol that adds RFs to Π, with main parties $\bar{\mathsf{P}}_i$ under party-wise malicious corruptions and specious subversions, and sub-parties $\bar{\mathsf{F}}_i$ under party-wise malicious corruptions. Let the corruption behaviors of $\bar{\mathsf{P}}_i \circ \bar{\mathsf{F}}_i$ translate from party-wise corruptions to the composed party corruptions following Table II. Let λ and r be the security parameter and randomness used by the environment $\mathcal{Z}$, respectively. If the RFs are anti-signalling and provide outer transparency, then for all PPT environments $\mathcal{Z}$, there exists a simulator $\mathcal{S}$ such that $\text{EXEC}_{\Pi,\mathcal{S},\mathcal{Z}}(\lambda, r) \approx \text{EXEC}_{\Pi',\mathcal{A},\mathcal{Z}}(\lambda, r)$, where $\mathcal{A}$ is a dummy adversary.

Proof: We adjust the proof of Theorem 2 in [23] to account for the differences mentioned above. Let π and π' denote the code of honest main parties running Π and Π', respectively.

In general, we should construct a simulator $\mathcal{S}_\Pi$ interacting with Π such that $\mathcal{Z}$ cannot distinguish an interaction with $\mathcal{S}_\Pi$ and Π from an interaction with $\mathcal{A}$ and Π'. $\mathcal{S}_\Pi$ simulates an execution of Π' internally. When interacting with an execution of protocol Π, $\mathcal{S}_\Pi$ can only send semi-honest or malicious corruption messages to main parties in Π. However, in the simulated execution of protocol Π', the main parties may be additionally under specious subversions. Therefore, $\mathcal{S}_\Pi$ should translate the corruptions of parties in Π' to the corruptions of parties in Π. The behaviors of $\mathcal{S}_\Pi$ in four primary corruption cases are shown in Fig. 7. The bold text highlights the main differences from the proof of Theorem 2 in [23]. The global subroutines γ are independent of $\mathcal{S}_\Pi$, $\mathcal{A}$ and $\mathcal{Z}$. Upon receiving a query from any party, $\mathcal{G}$ responds according to its own code. The indistinguishability argument is shown below.

Simulating honest $\bar{\mathsf{P}}_i$ and honest $\bar{\mathsf{F}}_i$: In this case, $\mathcal{S}_\Pi$ just forwards the messages output by P_i to $\bar{\mathsf{P}}_i$ as the simulated messages, and sends the messages received by $\bar{\mathsf{F}}_i$ to P_i. Since $\bar{\mathsf{F}}_i$ is of outer transparency, we have $\text{EXEC}_{\mathcal{Z},\Pi_1,\mathcal{A}}(\lambda, r) \approx \text{EXEC}_{\mathcal{Z},\Pi_2,\mathcal{A}}(\lambda, r)$, where $\mathcal{A}$ stands for the dummy adversary, Π_1 runs with no sub-parties and Π_2 runs with sub-parties. Obviously we have $\Pi_1 = \Pi, \Pi_2 = \Pi'$. Therefore, simulation of $\mathcal{S}_\Pi$ in this case is perfect.

Simulating honest $\bar{\mathsf{P}}_i$ and semi-honest $\bar{\mathsf{F}}_i$: The combination of honest $\bar{\mathsf{P}}_i$ and semi-honest $\bar{\mathsf{F}}_i$ in Π' is indistinguishable from *honest* P_i in Π. Firstly, no secret is leaked due to honest $\bar{\mathsf{P}}_i$. Moreover, although the randomness used by $\bar{\mathsf{F}}_i$ is revealed, the input to $\bar{\mathsf{P}}_i$ and the randomness used by $\bar{\mathsf{P}}_i$ are unknown to $\mathcal{A}$. Therefore, P_i remains honest, instead of semi-honest.

Case 1: honest $\bar{\mathsf{P}}_i$ and malicious $\bar{\mathsf{F}}_i$.
Upon receiving $(\mathtt{corr}, \mathsf{sid}, \bar{\mathsf{F}}_i, \mathsf{malicious})$ from $\mathcal{Z}$,
- **Send $(\mathtt{corr}, \mathsf{sid}, \mathsf{P}_i, \mathsf{malicious})$ to P_i and ignore further corruption messages.**
- **Upon receiving a feedback message from $\mathcal{Z}$ on behalf of $\bar{\mathsf{F}}_i$, run π and send the output to $\mathcal{Z}$ on behalf of $\bar{\mathsf{P}}_i$.**
- Upon receiving input to P_i in Π, run π and send the output to $\mathcal{Z}$ on behalf of $\bar{\mathsf{P}}_i$.
- Upon receiving a message (from an honest party) to P_i in Π, forward it to $\mathcal{Z}$.

Case 2: subverted $\bar{\mathsf{P}}_i$ and honest $\bar{\mathsf{F}}_i$.
Upon receiving $(\mathtt{corr}, \mathsf{sid}, \bar{\mathsf{P}}_i, (\mathtt{sub}, \bar{\pi}))$ from $\mathcal{Z}$,
- Record $(\mathsf{sid}, \bar{\mathsf{P}}_i, \mathtt{sub}, \bar{\pi})$ and ignore other corruptions.
- **Upon receiving output messages from P_i, forward them to $\mathcal{Z}$ on behalf of $\bar{\mathsf{F}}_i$.**
- Upon receiving further backdoor messages, send them to corresponding receiver.

Case 3: subverted $\bar{\mathsf{P}}_i$ and semi-honest $\bar{\mathsf{F}}_i$.
Upon receiving $(\mathtt{corr}, \mathsf{sid}, \bar{\mathsf{P}}_i, (\mathtt{sub}, \bar{\pi}))$ and $(\mathtt{corr}, \mathsf{sid}, \bar{\mathsf{F}}_i, \mathsf{semi\text{-}honest})$ from $\mathcal{Z}$,
- Record $(\mathsf{sid}, \bar{\mathsf{P}}_i, \mathtt{sub}, \bar{\pi})$ and ignore other corruptions.
- Send $(\mathtt{corr}, \mathsf{sid}, \mathsf{P}_i, \mathsf{semi\text{-}honest})$ to P_i.
- Upon receiving $(\mathsf{semi\text{-}honest}, \mathsf{sid}, mid, \mathsf{state})$ from P_i, run $\bar{\pi}$ and the code of $\bar{\mathsf{F}}_i$, send the updated state state' of $\bar{\mathsf{F}}_i$ to $\mathcal{Z}$ and the output message on behalf of $\bar{\mathsf{F}}_i$.
- Upon receiving "continue" from $\mathcal{Z}$ to $\bar{\mathsf{F}}_i$, forward it to P_i.
- Upon receiving further backdoor messages, send them to corresponding receiver.

Case 4: subverted $\bar{\mathsf{P}}_i$ and malicious $\bar{\mathsf{F}}_i$.
Upon receiving $(\mathtt{corr}, \mathsf{sid}, \bar{\mathsf{P}}_i, (\mathtt{sub}, \bar{\pi}))$ and $(\mathtt{corr}, \mathsf{sid}, \bar{\mathsf{F}}_i, \mathsf{malicious})$ from $\mathcal{Z}$,
- Record $(\mathsf{sid}, \bar{\mathsf{P}}_i, \mathsf{sub}, \bar{\pi})$ and ignore other corruptions.
- Send $(\mathtt{corr}, \mathsf{sid}, \mathsf{P}_i, \mathsf{malicious})$ to P_i.
- Upon receiving an input from $\mathcal{Z}$ to $\bar{\mathsf{P}}_i$, run $\bar{\pi}$ and forward the output message to $\mathcal{Z}$ on behalf of $\bar{\mathsf{P}}_i$. \\ Simulating behaviors due to new input.
- **Upon receiving a feedback message from $\mathcal{Z}$ on behalf of $\bar{\mathsf{F}}_i$, run $\bar{\pi}$ and send the output to $\mathcal{Z}$ on behalf of $\bar{\mathsf{P}}_i$.**
- Upon receiving a message (from an honest party) to P_i in Π, forward it to $\mathcal{Z}$.
- Upon receiving a message on behalf of $\bar{\mathsf{F}}_i$ to $\bar{\mathsf{P}}_i$, run $\bar{\pi}$ and send the output on behalf of P_i.

Fig. 7. The simulator for protocol Π'. Key differences from that in [23] are highlighted.

Simulating honest $\bar{\mathsf{P}}_i$ and malicious $\bar{\mathsf{F}}_i$: In this case, $\mathcal{A}$ is unable to obtain any secrets from $\bar{\mathsf{P}}_i$. Although $\bar{\mathsf{F}}_i$ may send "feedback" messages to $\bar{\mathsf{P}}_i$, thereby altering the underlying randomness, $\mathcal{A}$ still cannot extract any secret from $\bar{\mathsf{P}}_i$. However, since $\bar{\mathsf{F}}_i$ is malicious, it may sends arbitrary messages to other parties on behalf of P_i at its will. In this paper, we follow the standard corruption behavior for this case and treat the composed party as *malicious*. Therefore, $\mathcal{S}_\Pi$ can perfectly simulate honest $\bar{\mathsf{P}}_i$ in protocol Π'.

Simulating malicious $\bar{\mathsf{P}}_i$ and malicious/honest/semi-honest $\bar{\mathsf{F}}_i$: Since the main party $\bar{\mathsf{P}}_i$ in Π' is maliciously corrupted, no security is guaranteed for P_i in Π, whether $\bar{\mathsf{F}}_i$ is honest or not. Therefore, the combination of malicious $\bar{\mathsf{P}}_i$ and malicious/honest/semi-honest $\bar{\mathsf{F}}_i$ is indistinguishable from *malicious* P_i.

The corruption cases we are most interested in are those where a main party $\bar{\mathsf{P}}_i$ is subverted. There are three sub-cases, in which the firewall is honest, semi-honest and malicious, respectively.

Simulating subverted $\bar{\mathsf{P}}_i$ and honest $\bar{\mathsf{F}}_i$: By the anti-signalling property of $\bar{\mathsf{F}}_i$, P_i is still *honest:* Let $\widehat{\mathsf{P}}_i$ and $\widehat{\mathsf{F}}_i$ be the incorruptible version of $\bar{\mathsf{P}}_i$ and $\bar{\mathsf{F}}_i$, respectively. Let $\bar{\Pi}'$ be the same protocol with Π' except that $\bar{\mathsf{P}}_i \circ \bar{\mathsf{F}}_i$ is replaced by $\widehat{\mathsf{P}}_i \circ \widehat{\mathsf{F}}_i$. Since $\bar{\mathsf{F}}_i$ is anti-signalling, it holds that $1 - \Pr_{\mathsf{inf} \leftarrow I}[\text{EXEC}_{\Pi',\mathcal{A}(\mathsf{inf}),\mathcal{Z}}(\lambda, r) \approx$

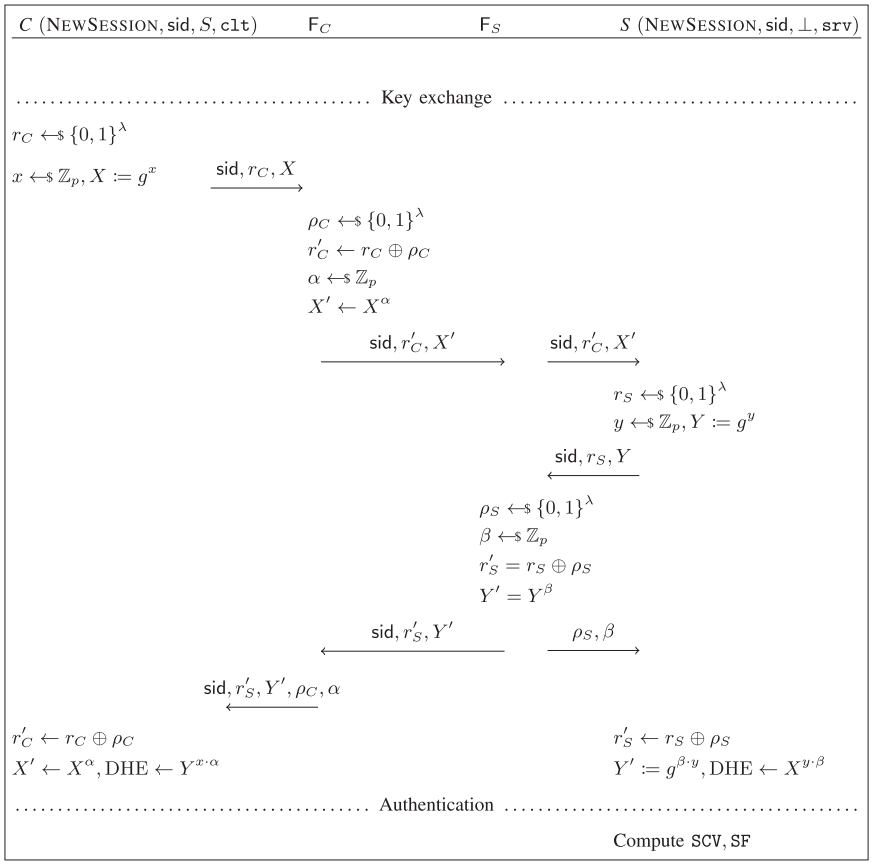


Fig. 8. Protocol srTLS (in the key exchange phase).

$\text{EXEC}_{\bar{\Pi}',\mathcal{A}(\text{inf}),\mathcal{Z}}(\lambda, r)]$ is negligible. Therefore, the environment $\mathcal{Z}$ cannot distinguish this case from the case where both $\bar{\mathsf{P}}_i$ and $\bar{\mathsf{F}}_i$ are honest, which is reduced to the first corruption case we have analyzed.

Simulating subverted $\bar{\mathsf{P}}_i$ *and semi-honest* $\bar{\mathsf{F}}_i$*:* In this case, the inputs to main parties are revealed due to the subversion corruption of $\bar{\Pi}$ and the inputs are revealed by a subverted implementation to $\bar{\mathsf{F}}_i$. Since the subversion is specious, the behaviors of the subverted main party are indistinguishable from an honest main party. Therefore, $\mathcal{S}_\Pi$ sends a *semi-honest* corruption message to P_i.

Simulating subverted $\bar{\mathsf{P}}_i$ *and malicious* $\bar{\mathsf{F}}_i$*:* The firewall is controlled by the adversary, which makes the "message sanitization" invalid. Therefore, $\mathcal{S}_\Pi$ does malicious corruption of P_i. The composed party of subverted $\bar{\mathsf{P}}_i$ and malicious $\bar{\mathsf{F}}_i$ is indistinguishable from *malicious* P_i. □

V. Strengthening Π_{uaKE} With Subversion Resilience

A. RFs for Π_{uaKE}

High Level Ideas: To resist the subversion attacks, we construct RFs for both clients and servers. The core idea is to "sanitize" the DH key exchange part in the handshake phase, including re-randomizing the ephemeral keys and nonces. Constructing RFs with strong transparency for the TLS 1.3 handshake protocol is challenging due to the protocol's resilience to various forms of man-in-the-middle attacks in a strong security model. Consequently, the transparency we consider in this paper is outer transparency, in which a main party is aware of the existence of its RF. In this context, an RF is permitted to return feedback messages (utilized for re-randomizing the output messages) to the party.

Building on this insight, we have developed firewalls for both the client and server, as depicted in Fig. 8, where F_C and F_S represent RFs for the client C and server S, respectively. Note that we focus on constructing RFs for the key exchange phase in Fig. 3. Therefore, most messages in the authentication phase are not shown in Fig. 8.

Initially, a client C generates a `ClientHello` message, containing two core elements for key exchange: r_C and X. F_C re-randomizes them using ρ_C and α. Subsequently, F_C transmits (sid, r'_C, X') to the server. On the server side, F_S forwards (sid, r'_C, X') to S. Upon receiving (sid, r_S, Y) from S, F_S selects randomness ρ_S and β to re-randomize r_S, Y and gets r'_S and Y'. F_S then sends (sid, r'_S, Y') to C and returns the randomness for reradomization to S. Subsequently, S computes r'_S, Y' and generates a signature and a MAC. During the authentication phase, F_S forwards the messages sent by S. On the client side, F_C forwards r'_S and Y' to C, along with the randomness used by F_C. Then C updates r_C and X. Subsequently, both parties continue to follow the RFC specification in the remaining steps.

The subversion-resilient signatures: When we instantiate the certification functionality $\mathcal{G}_{\mathsf{cert}}^{S}$ with a signature scheme in TLS 1.3, we should consider the problem of subversion attacks against signature schemes. TLS 1.3 specification provides four alternative signature schemes used for handshake protocol (i.e., ECDSA [46], RSASSA-PSS RSAE [47], EdDSA [48] and RSASSA-PSS PSS [47] algorithms), two of which can be deterministic, i.e., the EdDSA algorithms [48] and the deterministic ECDSA algorithms [49], which are naturally subversion-resilient.

B. Realizing $\mathcal{F}_{\mathsf{uaKE}}$ Under Specious Subversions

The unilaterally authenticated key exchange protocol srTLS is shown in Fig. 8. Based on Theorem 9, we can prove the security of srTLS in the plain UC model.

Theorem 10: srTLS UC-realizes $\mathcal{F}_{\mathsf{uaKE}}$ in the presence of global subroutines $(\mathcal{G}_{\mathsf{cert}}^{S}, \mathcal{G}_{\mathsf{bb}})$ with main parties $\mathcal{P}$ under specious subversions or malicious corruptions, and sub-parties $\mathsf{F}_{\mathcal{P}}$ being honest, semi-honest or malicious.

Proof: In Theorem 8, we have proved that Π_{uaKE} UC-realizes $\mathcal{F}_{\mathsf{uaKE}}$ in the presence of global functionalities $(\mathcal{G}_{\mathsf{cert}}^{S}, \mathcal{G}_{\mathsf{bb}})$. By Theorem 9, it remains to prove that the constructions of RFs for Π_{uaKE} provide outer transparency and anti-signalling property.

Lemma 11: The RFs in Fig. 8 are of outer transparency and anti-signalling.

Proof. Outer transparency: On the server-side, it is easy to see that F_S do not change the format of `ServerHello`. Moreover, the randomness used for re-randomizing r_S, Y (i.e., ρ_S and β) are uniformly chosen. Assuming that S and F_S are both incorruptible, then r_S and Y are uniformly random. Then the modified values r'_S and Y' remain uniformly random. The behavior of the composed party $S \circ \mathsf{F}_S$ in protocol srTLS is indistinguishable from that of S in protocol Π_{uaKE}.

Formally, let $\widetilde{S}$ and $\widetilde{\mathsf{F}}_S$ be incorruptible versions of S and F_S, respectively. Let Π be the protocol Π_{uaKE} run by $\widetilde{S}$ and an incorruptible client C. Let Π' be a protocol identical to Π except that the server $\widetilde{S}$ is installed with $\widetilde{\mathsf{F}}_S$. For all PPT environments $\mathcal{E}$, it holds that

$$\text{EXEC}_{\mathcal{E},\Pi,\mathcal{A}}(\lambda, r) \approx \text{EXEC}_{\mathcal{E},\Pi',\mathcal{A}}(\lambda, r),$$

where $\mathcal{A}$ stands for the dummy adversary. By a similar argument, F_C is also of outer transparency.

Anti-signalling: First we show that F_S is anti-signalling. Let $\bar{S}$ be a subverted version of S. Let $\widehat{S}$ and $\widehat{\mathsf{F}}_S$ be the incorruptible version of S and F_S, respectively. Let Π be the protocol Π_{uaKE} run by $\bar{S} \circ \widehat{\mathsf{F}}_S$, and Π' be the same protocol with Π except that $\bar{S}$ is replaced by $\widehat{S}$. For a subverted server $\bar{S}$, r_S and Y are probably biased due to the subverted code $\hat{\pi}$. However, $\widehat{\mathsf{F}}_S$ guarantees that the updated values $r'_S := r_S \oplus \rho_S$ and $Y' := Y^{\beta}$, where ρ_S and β are randomness generated by $\widehat{\mathsf{F}}_S$, are always uniformly distributed. The other authentication messages, including `Cert`, `ServerCertVrf` and `ServerFinished`, are deterministic once srv_random and srv_key_share are specified. Therefore, an adversary cannot extract any secret information from these handshake messages. It is easy to see that for all PPT environments $\mathcal{Z}$, it holds that $1 - \Pr_{\mathsf{inf} \leftarrow I}[\text{EXEC}_{\Pi,\mathcal{A}(\mathsf{inf}),\mathcal{Z}}(\lambda, r) \approx \text{EXEC}_{\Pi',\mathcal{A}(\mathsf{inf}),\mathcal{Z}}(\lambda, r)]$ is negligible, where $\mathcal{A}$ is a dummy adversary which has access to the additional information inf embedded in the subverted code $\bar{\pi}$. An analogous analysis can be performed on the client side: even though r_C and Y may be biased due to subverted implementation, an honest RF F_C guarantees the uniformly random distribution of r_C and Y. Furthermore, the derived `ClientFinished` message does not exfiltrate secret information of the client to the adversaries. Therefore, the RF F_C for the client is also anti-signalling. □

The theorem follows immediately. □

TABLE V
EFFICIENCY COMPARISON BETWEEN SCHEMES, WHERE exp. MEANS EXPONENTIATION OPERATION IN GROUP $\mathbb{G}$, PAIRING REFERS TO A BILINEAR PAIRING OPERATION

Schemes		DMS [9]	BBFOM [15]	**Ours**
Client	Party	$3 \times$ exp. + $1 \times$ pairing	$4 \times$ exp.	$2 \times$ exp.
	Firewall	$4 \times$ exp.	$5 \times$ exp.	$1 \times$ exp.
Server	Party	$4 \times$ exp. + $1 \times$ pairing	$5 \times$ exp.	$3 \times$ exp.
	Firewall	$1 \times$ exp.	-	$1 \times$ exp.
Communication		$3 \times \mathbb{G} + 2 \times \mathbb{Z}_p$	$6 \times \mathbb{G} + 2 \times \mathbb{Z}_p$	$2 \times \mathbb{G}$

VI. PERFORMANCE EVALUATIONS

We present an analysis of the theoretical complexity of our protocol and provide a detailed evaluation of its concrete running time based on our implementation.

A. Theoretical Analysis

We first conduct a theoretical analysis of our protocol's efficiency in terms of computational and communication costs, as shown in Table V. Note that the cost of signature generation and verification is not included in the table. The primary computational cost arises from exponentiations and bilinear pairings. In the DMS protocol [9], the Pedersen commitment adds two more exponentiations than the plain DH key exchange. The firewall for the client also requires 4 exponentiations to maul and re-randomize the commitment. On the server side, verification of the commitment costs 2 additional exponentiations. Both parties are required to perform one bilinear pairing operation.

The BBFOM protocol [15] requires a public key encryption scheme, instantiated with ElGamal, which adds two exponentiations on the client side. The firewall for a client requires two exponentiations for re-randomization, one for decryption, and two for re-encryption. Evidently, our scheme incurs significantly lower costs for both parties and firewalls compared to the other two schemes.

Regarding the communication cost, we only consider the messages transmitted in the key exchange phase, excluding the encrypted handshake messages such as `ServerCertVrf`, `ServerFinished`, and `ClientFinished`.

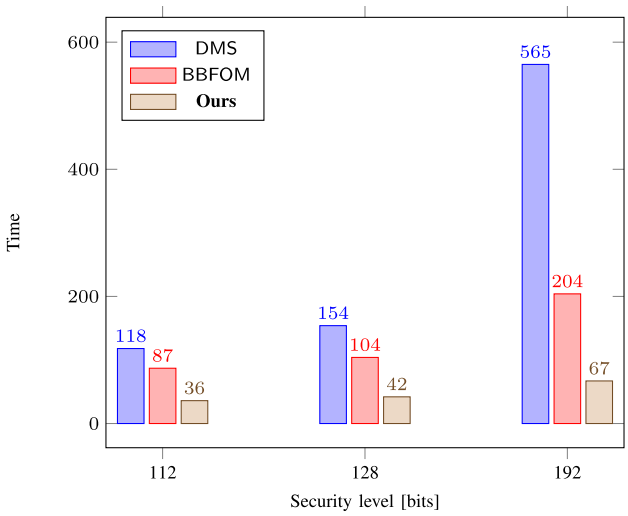


Fig. 9. The time cost of protocol executions without RFs (/ms).

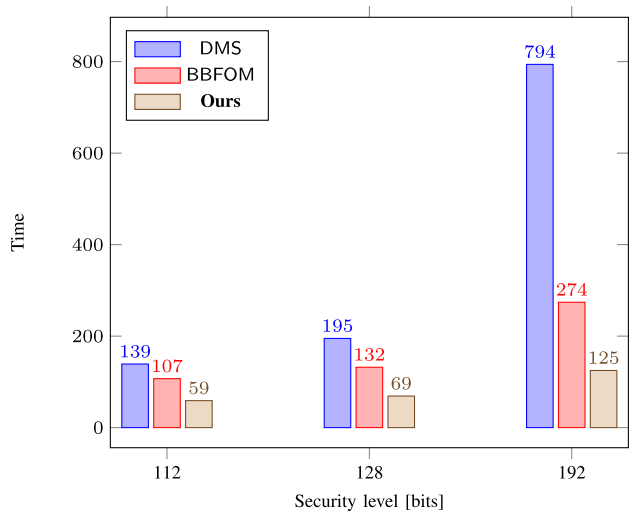


Fig. 10. The time cost of protocol executions with RFs. (/ms).

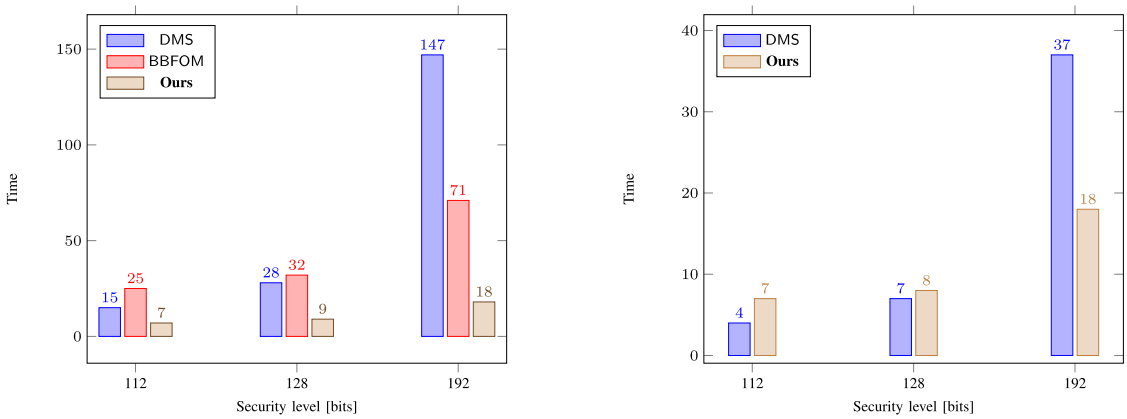
B. Experimental Analysis

In this section, we compare the implementation of our protocol with DMS [9] and BBFOM [15]. All schemes are implemented using the charm-crypto and pypbc libraries. The elliptic curve parameters recommended by [50] are referenced in accordance with varying security strength levels. Frequently employed security strengths include 80, 112, 128 and 192 bits, with a security strength of 80 bits no longer deemed sufficient. To conduct a comparative analysis of our scheme against two other schemes across different security levels, we have employed three curves for our scheme and BBFOM [15]: secp224r1, secp256r1 and secp384r1, each representing security levels of 112, 128 and 192 bits, respectively. Additionally, ss1024, ss1536 and ss3840 curves have been utilized for the implementation of the DMS scheme, with the prefix ss denoting *supersingular* curves followed by the number of bits in a prime number p. These sizes are set to align with the standard levels of bit security [51] as per RFC 5091 [52]. The latter document details the correlation between NIST SP 800-57 security levels and sizes for supersingular curves. Notably, in the DMS scheme, both elements belong to the same group, necessitating pairings in DMS to be carried out over supersingular curves. Consequently, the available options for implementing DMS are limited.

We have adapted all three schemes to the TLS 1.3 handshake protocol. To ensure compatibility with the implementation of TLS 1.3, we have made specific modifications to the specifications of both DMS and BBFOM. In the DMS implementation, both parties sign the transcript instead of the bilinear pairings. As a consequence, RFs for both servers and clients are required to send feedback messages to their respective parties. To elaborate further, the RF for the client sends randomness used for re-randomization and mauling operations back to the client. Similarly, the RF for the server sends randomness for mauling operations back to the server. Moreover, the RFs are not required to verify the signature over bilinear pairings due to the encryption of the signatures. The main effect of these modifications is that the bilinear pairing computation of a party is replaced with an exponentiation operation, and the bilinear pairing computation of RFs is removed. It implies that the adapted version of DMS needs one more exponentiation operation but two less bilinear pairings for one party and its RF. Based on our experiments, the time required for one exponentiation operation is 2.8 ms on elliptic curve ss1024 , 4.9ms on ss1536 , and 25.3ms on ss3840 . In comparison, the time for one bilinear pairing operation is 2.6ms on ss1024 , 5.7ms on ss1536 , and 57.1ms on ss3840. It indicates that performing a single exponentiation operation costs significantly less than computing bilinear pairings twice. Hence, the adapted version of DMS clearly reduces computational costs compared to the original protocol. The handshake protocol of BBFOM follows a TLS 1.3-like protocol, as depicted in Fig. 5 of [53], with the exception that we utilize PKCS#1 RSA encryption for the encryption of c in the `ClientHello` message. As the `ServerCertVrf` is encrypted by the handshake secret, RFs are unable to sanitize or even verify the signatures. Consequently, in our implementations, the reverse firewall does not verify the signatures. For the sake of brevity, we continue to use DMS and BBFOM to represent the adapted protocols.

Time Cost: Now we look into concrete time cost of our scheme in comparison with previous work. To the best of our knowledge, Bossuat et al. [15] are the only ones to estimate the performance of RFs for an AKE protocol. In the following experiments, we measure the execution time of a single handshake, which refers to the duration from initiating the handshake protocol by the client until both parties are able to compute the session keys. Our measurements were conducted on an 8-core Intel(R) Core(TM) i7-9750H CPU running at 2.6 GHz within a virtual machine environment. Each protocol undergoes 1000 executions for every security bit level. The network bandwidth is 100 Mbps. The average network latency is 0.02 ms.

Protocol executions without RFs: In the absence of RFs, messages are forwarded honestly. Our proposed scheme aligns with the original TLS 1.3 handshake protocol, yielding significantly faster execution time compared to the other two schemes. The



(a) The time cost of RFs for the clients.

(b) The time cost of RFs for the servers.

Fig. 11. Time cost of the RFs (/ms).

evaluation result is shown in Fig. 9. The time cost reduction ranges from 58.62% (112-bit security, compared to the BBFOM scheme) to 88.14% (192-bit security, compared to the DMS scheme).

Protocol executions with RFs: Fig. 10 presents the total execution time of the schemes when both parties are equipped with RFs that modify the messages output by parties. The result shown in Fig. 10 reveals that even when firewalls for both parties work, the execution times for DMS and BBFOM remain significantly longer than those of our scheme. Specifically, our scheme exhibits a total time cost that is 57.55% to 84.26% lower than that of DMS, and 44.86% to 54.38% lower than that of BBFOM. For the DMS scheme, the time cost for the 192-bit security is significantly higher than that for the other two security levels. This discrepancy arises from the selection of supersingular curves for achieving 192-bit security, where the time cost of exponentiation operations is notably higher compared to the supersingular curves chosen for 112-bit and 128-bit security levels.

The time cost of RFs for the clients and the servers is shown in Fig. 11. Note that the BBFOM scheme has no RFs for the servers. Furthermore, in the adapted DMS scheme applied to TLS 1.3, it is observed that the time cost associated with RFs for clients is notably higher compared to that for servers. This discrepancy stems from the fact that a client's RF necessitates additional operations involving re-randomization and manipulation of commitments sent by the client, whereas a server's RF solely requires manipulation of a key share transmitted by the server. Consequently, the number of exponentiation operations performed by a client's RF surpasses those executed by a server's RF.

ACKNOWLEDGMENT

We would like to thank all anonymous reviewers for their valuable comments.

REFERENCES

[1] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," *RFC*, vol. 8446, pp. 1–160, 2018, doi: 10.17487/RFC8446.

[2] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A cryptographic analysis of the TLS 1.3 handshake protocol," *J. Cryptol.*, vol. 34, no. 4, Oct. 2021, Art. no. 37.

[3] D. Diemert and T. Jager, "On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments," *J. Cryptol.*, vol. 34, no. 3, Jul. 2021, Art. no. 30.

[4] J. Hesse, S. Jarecki, H. Krawczyk, and C. Wood, "Password-authenticated TLS via OPAQUE and post-handshake authentication," in *Proc. Adv. Cryptol.*, Lyon, France, Springer, Apr. 23–27, 2023, pp. 98–127.

[5] D. Shumow and N. Ferguson, "On the possibility of a backdoor in the NIST SP800–90 dual EC PRNG," CRYPTO 2007 Rump Session. 2007. [Online]. Available: http://rump2007.cr.yp.to/15-shumow.pdf

[6] S. Checkoway et al., "On the practical exploitability of dual EC in TLS implementations," in *Proc. 23rd USENIX Secur. Symp.*, San Diego, CA, USA: USENIX Association, Aug. 20–22, 2014, pp. 319–335.

[7] I. Mironov and N. Stephens-Davidowitz, "Cryptographic reverse firewalls," in *Proc. Adv. Cryptol.*, Sofia, Bulgaria, Springer, Apr. 26–30, 2015, pp. 657–686.

[8] G. Ateniese, B. Magri, and D. Venturi, "Subversion-resilient signature schemes," in *Proc. 22nd Conf. Comput. Commun. Secur.*, Denver, CO, USA: ACM Press, Oct. 12–16, 2015, pp. 364–375.

[9] Y. Dodis, I. Mironov, and N. Stephens-Davidowitz, "Message transmission with reverse firewalls—secure communication on corrupted machines," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 14–18, 2016, pp. 341–372.

[10] S. Chakraborty, S. Dziembowski, and J. B. Nielsen, "Reverse firewalls for actively secure MPCs," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 17–21, 2020, pp. 732–762.

[11] C. Ganesh, B. Magri, and D. Venturi, "Cryptographic reverse firewalls for interactive proof systems," in *Proc. 47th Int. Colloq. Automata Lang. Program.*, Saarbrücken, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Jul. 8–11, 2020, pp. 55:1–55:16.

[12] S. Chakraborty, C. Ganesh, M. Pancholi, and P. Sarkar, "Reverse firewalls for adaptively secure MPC without setup," in *Proc. Adv. Cryptol.*, Singapore: Springer, Dec. 6–10, 2021, pp. 335–364.

[13] S. Chakraborty, B. Magri, J. B. Nielsen, and D. Venturi, "Universally composable subversion-resilient cryptography," in *Proc. Adv. Cryptol.*, Trondheim, Norway: Springer, May 30–Jun. 3, 2022, pp. 272–302.

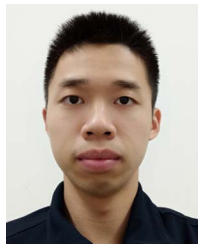
[14] S. Chakraborty, C. Ganesh, and P. Sarkar, "Reverse firewalls for oblivious transfer extension and applications to zero-knowledge," in *Proc. Adv. Cryptol.*, Lyon, France, Springer, Apr. 23–27, 2023, pp. 239–270.

[15] A. Bossuat, X. Bultel, P.-A. Fouque, C. Onete, and T. van der Merwe, "Designing reverse firewalls for the real world," in *Proc. 25th Eur. Symp. Res. Comput. Secur.*, Guildford, UK: Springer, Sep. 14–18, 2020, pp. 193–213.

[16] M. Bellare, K. G. Paterson, and P. Rogaway, "Security of symmetric encryption against mass surveillance," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 17–21, 2014, pp. 1–19.

[17] M. Armour and B. Poettering, "Subverting decryption in AEAD," in *Proc. 17th IMA Int. Conf. Cryptogr. Coding*, Oxford, U.K.: Springer, Dec. 16–18, 2019, pp. 22–41.

[18] H. Davis, D. Diemert, F. Günther, and T. Jager, "On the concrete security of TLS 1.3 PSK mode," in *Proc. Adv. Cryptol.*, Trondheim, Norway: Springer, May 30–Jun. 3, 2022, pp. 876–906.
[19] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proc. Adv. Cryptol.*, Amsterdam, The Netherlands: Springer, Apr. 28–May 2, 2002, pp. 337–351.
[20] A. Groce and J. Katz, "A new framework for efficient password-based authenticated key exchange," in *Proc. 17th Conf. Comput. Commun. Secur.*, Chicago, Illinois, USA: ACM Press, Oct. 4–8, 2010, pp. 516–525.
[21] C. Badertscher, R. Canetti, J. Hesse, B. Tackmann, and V. Zikas, "Universal composition with global subroutines: Capturing global setup within plain UC," in *Proc. 18th Theory Cryptogr. Conf.*, Durham, NC, USA: Springer, Nov. 16–19, 2020, pp. 1–30.
[22] R. Canetti, D. Shahaf, and M. Vald, "Universally composable authentication and key-exchange with global PKI," in *Proc. 19th Int. Conf. Theory Pract. Public Key Cryptogr.*, Taipei, Taiwan: Springer, Mar. 6–9, 2016, pp. 265–296.
[23] P. Arnold, S. Berndt, J. Müller-Quade, and A. Ottenhues, "Protection against subversion corruptions via reverse firewalls in the plain universal composability framework," Cryptology ePrint Archive, Paper 2023/1951, 2023. [Online]. Available: https://eprint.iacr.org/2023/1951
[24] A. Young and M. Yung, "The dark side of "black-box" cryptography, or: Should we trust capstone?," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 18–22, 1996, pp. 89–103.
[25] A. Young and M. Yung, "Kleptography: Using cryptography against cryptography," in *Proc. Adv. Cryptol.*, vol. 1233. Konstanz, Germany: Springer, May 11–15, 1997, pp. 62–74.
[26] M. Bellare, J. Jaeger, and D. Kane, "Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks," in *Proc. 22nd Conf. Comput. Commun. Secur.*, Denver, CO, USA: ACM Press, Oct. 12–16, 2015, pp. 1431–1440.
[27] J. P. Degabriele, P. Farshim, and B. Poettering, "A more cautious approach to security against mass surveillance," in *Fast Software Encryption – FSE 2015*, G. Leander, Ed. Istanbul, Turkey: Springer, Mar. 8–11, 2015, pp. 579–598.
[28] R. Chen, X. Huang, and M. Yung, "Subvert KEM to break DEM: Practical algorithm-substitution attacks on public-key encryption," in *Proc. Adv. Cryptol.*, Daejeon, South Korea: Springer, Dec. 7–11, 2020, pp. 98–128.
[29] S. Berndt, J. Wichelmann, C. Pott, T.-H. Traving, and T. Eisenbarth, "ASAP: Algorithm substitution attacks on cryptographic protocols," in *Proc. 17th ACM Symp. Inf. Comput. Commun. Secur.*, Nagasaki, Japan: ACM Press, May 30 – Jun. 3, 2022, pp. 712–726.
[30] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Generic semantic security against a kleptographic adversary," in *Proc. 24th Conf. Comput. Commun. Secur.*, Dallas, TX, USA: ACM Press, Oct. 31–Nov. 2, 2017, pp. 907–922.
[31] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Correcting subverted random oracles," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 19–23, 2018, pp. 241–271.
[32] A. Russell, Q. Tang, M. Yung, and H.-S. Zhou, "Cliptography: Clipping the power of kleptographic attacks," in *Proc. Adv. Cryptol.*, Hanoi, Vietnam: Springer, Dec. 4–8, 2016, pp. 34–64.
[33] M. Fischlin and S. Mazaheri, "Self-guarding cryptographic protocols against algorithm substitution attacks," in *Proc. IEEE 31st Comput. Secur. Found.s Symp.*, Oxford, UK: IEEE Computer Society Press, Jul. 9–12, 2018, pp. 76–90.
[34] M. Fischlin, "Stealth key exchange and confined access to the record protocol data in TLS 1.3," in *Proc. 30th Conf. Comput. Commun. Secur.*, 2023, pp. 2901–2914.
[35] S. Chakraborty, L. Magliocco, B. Magri, and D. Venturi, "Key exchange in the post-snowden era: Universally composable subversion-resilient PAKE," in *Proc. Adv. Cryptol.*, Berlin, Germany: Springer, Dec. 7–11, 2024.
[36] R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, and H. Wee, "Efficient password authenticated key exchange via oblivious transfer," in *Proc. 15th Int. Conf. Theory Pract. Public Key Cryptogr.*, May 21–23, 2012, pp. 449–466.
[37] D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan, "Key homomorphic PRFs and their applications," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 18–22, 2013, pp. 410–428.
[38] J. Katz and Y. Lindell, *Introduction to Modern Cryptography: Principles and Protocols*, Boca Raton, FL, USA: Chapman and Hall/CRC, 2007.
[39] H. Krawczyk and P. Eronen, "HMAC-based extract-and-expand key derivation function (HKDF)," *RFC*, vol. 5869, pp. 1–14, 2010, doi: 10.17487/RFC5869.
[40] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," *RFC*, vol. 2104, pp. 1–11, 1997, doi: 10.17487/RFC2104.
[41] M. Bellare and V. T. Hoang, "Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model," in *Proc. Adv. Cryptol.*, Sofia, Bulgaria: Springer, Apr. 26–30, 2015, pp. 627–656.
[42] R. Canetti, "Universally composable signature, certification, and authentication," in *Proc. 17th IEEE Comput. Secur. Foundations Workshop*, 2004, pp. 219–233.
[43] Y. Dodis, J. Katz, A. Smith, and S. Walfish, "Composability and on-line deniability of authentication," in *Proc. 6th Theory Cryptogr. Conf.*, Berlin, Germany: Springer, Mar. 15–17, 2009, pp. 146–162.
[44] R. Canetti, P. Jain, M. Swanberg, and M. Varia, "Universally composable end-to-end secure messaging," in *Proc. Adv. Cryptol.*, Santa Barbara, CA, USA: Springer, Aug. 15–18, 2022, pp. 3–33.
[45] R. Canetti, "Universally composable security," *J. ACM*, vol. 67, no. 5, pp. 28:1–28:94, 2020, doi: 10.1145/3402457.
[46] American National Standards Institute, Inc., "ANSI X9.62 public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ECDSA)," Nov. 16, 2005. [Online]. Available: https://standards.globalspec.com/std/1955141/ANSI%20X9.62
[47] K. M. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA cryptography specifications version 2.2," *RFC*, vol. 8017, pp. 1–78, 2016, doi: 10.17487/RFC8017.
[48] S. Josefsson and I. Liusvaara, "Edwards-curve digital signature algorithm (EdDSA)," *RFC*, vol. 8032, pp. 1–60, 2017, doi: 10.17487/RFC8032.
[49] T. Pornin, "Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA)," *RFC*, vol. 6979, pp. 1–79, 2013, doi: 10.17487/RFC6979.
[50] L. Chen, D. Moody, K. Randall, A. Regenscheid, and A. Robinson, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," 2023. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/186/final
[51] E. Barker, "SP 800–57. recommendation for key management: Part 1 - general," 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-57pt1r5
[52] X. Boyen and L. Martin, "Identity-based cryptography standard (IBCS) #1: Supersingular curve implementations of the BF and BB1 cryptosystems," *RFC*, vol. 5091, pp. 1–63, 2007, doi: 10.17487/RFC5091.
[53] A. Bossuat, X. Bultel, P.-A. Fouque, C. Onete, and T. van der Merwe, "Designing reverse firewalls for the real world," Cryptology ePrint Archive, Report 2020/854, 2020, https://eprint.iacr.org/2020/854



Jiahao Liu received the master's degree from the National University of Defense Technology, China, in 2020. He is currently working toward the PhD degree with the National University of Defense Technology. His research interest includes network security and public key cryptography.

Yi Wang received the PhD degree from the National University of Defense Technology, China, in 2021. He is currently a research associate with the College of Computer Science and Technology, National University of Defense Technology, China. His research interests include public-key cryptography, information security and cyber security. He was awarded the Young Elite Scientists Sponsorship by China Association for Science and Technology, in 2023.

Xincheng Tang received the BS degree from the National University of Defense Technology. He is currently working toward the master's degree with the National University of Defense Technology. His research interests include cryptography and network security.

Rongmao Chen received the PhD degree in computer science from the University of Wollongong, Australia. He is now an associate professor with the College of Computer Science and Technology, National University of Defense Technology in China. His major research interests include applied cryptography and cybersecurity. He has published more than 50 research papers in refereed international conferences and journals, such as CRYPTO, ASIACRYPT, PKC, and CT-RSA. He has served as the program committee member on various international conferences such as ASIACRYPT, ACM CCS, and PKC. He received the prestigious Young Elite Scientists Sponsorship by CAST (2017), Rising Star Award by ACM SIGSAC CHINA (2020), and Distinguished Young Scholars by National Natural Science Foundation of China (2021).

Xinyi Huang received the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently a professor with the College of Cyber Security, Jinan University, China. He has authored more than 100 research papers in refereed international conferences and journals. His work has been cited more than 13000 times at Google Scholar (H-index: 59). His research interests include applied cryptography and network security.

Jinshu Su (Senior Member, IEEE) received the BS degree in mathematics from Nankai University, Tianjin, in 1985, and the MS and PhD degrees in computer science from the National University of Defense Technology, Changsha, in 1988 and 2000, respectively. He is a professor with the Academy of Military Science. His research interests include Internet architecture, network security, and AI for networking.