IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 11, 2019, accepted July 23, 2019, date of publication July 26, 2019, date of current version August 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931472

End-to-End Authenticated Key Exchange Based on Different Physical Unclonable Functions

JIN WOOK BYUN, (Member, IEEE)
Department of Information and Communication, Pyeongtaek University, Pyeongtaek-si 17869, South Korea
e-mail: jwbyun@ptu.ac.kr

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03032424).

ABSTRACT We propose the first authenticated key exchange (AKE) protocol with different physical unclonable functions (PUFs). Our protocol allows for two end-users each holding a distinct PUF-embedded device and a long-term secret to agree to an authenticated session key. For malicious behaviors on the PUF-embedded device, we first define a Device query, which models an adversary who intentionally (or unintentionally) picks up an arbitrary device and attempts to input a message and obtain an output. As per the author's knowledge, this is the first study to consider Corrupt queries that return long-term secrets in the PUF-embedded device and its relevant platform. In this paper, we prove that our protocol is secure under a new security model and requires three flows to achieve a secure AKE with users' distinct PUFs. As it requires no intervention by a central server after its initial setup phase, it is suitable for practical decentralized networks.

INDEX TERMS Physical unclonable function, authentication, device authentication, authenticated key exchange.

I. INTRODUCTION

A physical unclonable function (PUF) is physically embodied in the device hardware. It always returns an unpredictable output depending on random physical factors, which are generally initialized at the manufacturing stage. Such physical factors are assumed to be uniquely assigned to each PUF, which cannot be cloned or identically remanufactured. Any malicious attempt to reproduce or clone the function changes its hardware structure and renders it useless. Owing to its unpredictability and unclonability, the PUF has been widely utilized in cryptography for enhancing security. For instance, in conventional public key infrastructure PKI-based banking settings, users secretly manage their long-term private keys in a nonvolatile memory and utilize them later for authentication or secure transaction. One critical weakness of such a setup is that it cannot completely be free from a long-term private-key-exposure attack. In contrast, in the PUF-based setting, the PUF embodied in a device first receives a long-term secret and always outputs a noisy result, with increased entropy, and which is completely unpredictable against any adversary who already obtains the long-term secrets. For this reason, the PUF has been considered as a countermeasure to prevent a key-exposure attack. In the last decade, excellent studies have been conducted on PUFs to securely and efficiently combine the existing cryptographic primitives. They created cryptographic primitives based on PUFs, for example oblivious transfer (OT) [33], bit commitment (BC) [33], cryptographic key or random number generator [34], [41], authenticated encryption [2], authentication or authenticated key exchange (AKE) [9], [14]. As such, PUF-based primitives have become an essential technique in cryptology.

The current study focused on a PUF-based AKE, in which the subject of changeable network requirements has always been a big issue in cryptology. In particular, whenever a new innovative technology (e.g., PUF) emerges, a new methodology on efficient and secure construction for the AKE should be re-examined under a new security model. Until now, interestingly, most PUF-based AKE schemes between two entities (server and user) have assumed a single PUF owned by a user (or a server). That is, a user (or a server) first registers an output of a PUF in the registration phase, and then the user (or a server) recovers the secret and performs an AKE protocol in the authentication phase. In cryptology, this has been a typical way of use of a single PUF between a server and user. However, under a single-PUF setting, a server's

The associate editor coordinating the review of this manuscript and approving it for publication was Mamoun Alazab.

database (DB) cannot basically support an end-to-end (or device-to-device) authentication that are naturally based on different PUFs. In addition, its DB should be securely maintained with no small cost. Thus, the single PUF scenario is not suitable for the decentralized network such as IoT, P2P setting. Instead, a different PUF setting in which a PUF-embedded device makes a secure channel with another PUF-embedded device through a mutual authentication is more suitable for the decentralized scenario.

In our work, we focused on a practical PUF-based end-to-end setting in which two users, who usually have distinct PUF-embedded devices, execute an AKE protocol without any help from the server. Our goal was to construct an AKE in which only valid users, possessing a unique PUF-embedded device and a long-term secret, can successfully authenticate each other by using a mutually agreed upon session key. However, owing to the use of two distinct PUFs, the establishment of a common keying material during an authentication protocol is not easy. This can be inevitably solved by making full use of the registration phase. Thus, one of the important tasks is to determine initial values and how they should be setup in the registration phase. Based on the initial setting, two users recover the same secrets and perform the usual AKE protocols. In this paper, we present a proper security model based on a previous model [10] and construct different PUF-based AKEs. The first construction has a structural weakness and explicitly shows the importance of the appropriate design of the registration phase. The second PUF-based AKE is the final version that fully achieves our security goal.

II. RELATED WORKS AND OUR CONTRIBUTION

A. RELATED WORKS

Over the last decade, extensive research has been conducted in areas of PUF-based authentication or AKE under various settings (e.g., IoT, RFID, smart meter system, and smart grid) [1], [5], [6], [8], [11], [12], [17], [18], [24], [25], [27], [29]–[32], [37], [39], [44], [45]. A PUF has been applied in an RFID authentication for enhancing security and efficiency. Sadeghi *et al.* [37] first applied a PUF in RFID authentication to guarantee destructive privacy, i.e., privacy of a tag, which has been originally introduced by the Vaudenay security model, is destroyed after its corruption [43]. The protocol is simple in that a tag evaluates a PUF with its secret status S then derives an authentication key K for authentication [37]. Then, a new destructive private PUF-based authentication protocol was designed for a large-scale RFID system [1], and it guarantees scalability that a server does not need a linear search operation to identify tags. The protocol provides mutual authentication with three communication flows, and its security is also analyzed using the Vaudenay security model. In 2010, Kulseng *et al.* [27] proposed a PUF-based lightweight mutual verification protocol that uses a PUF and linear feedback shift register for low-cost RFID tag. It consists of four rounds and encrypts ($\oplus$ operation) a tag's ID by using a shared secret key. For every session, the secret key is updated using the PUF on the tag side. However, its security was reexamined and enhanced in view of ID protection (confidentiality) and desynchronized attack [24], [44]. In 2013, Moriyama et al. [32] presented a novel PUF-based RFID authentication protocol (MMY) under a new tag privacy model that supports secret-key leakages in nonvolatile memory (NVM). The MMY allows a RFID reader to update random status secret s and then a tag receives s for each session; this is the main concept for ensuring security against key leakage in NVM. In 2015, a new state-aware privacy model was suggested in an RFID setting [29]. The model classifies the existing RFID protocols into stateful and stateless authentication protocols, depending on the existence of updatable secrets. A new stateful-private RFID authentication protocol was constructed considering the unpredictability of PUF. However, its construction is based on a secure public-key encryption. In 2017, a combination PUF with a threshold cryptography was presented in an RFID system. It uses a secret sharing technique for thwarting tag compromising attacks [23].

In 2017, Chatterjee et al. have proposed a PUF-based AKE between nodes under IoT setting [11]. The protocol has been designed under a three party setting where two nodes do AKE by the help of a server. However, it has been flawed by Braeken et al. in 2018 [4]. Braeken has shown that the protocol [11] is weak against man-in-the-middle, impersonation, and replay attacks [4]. In 2019, Chatterjee et.al. have proposed PUF-based AKE without explicit PUF's challenge and response pairs (CRP) under IoT setting [12]. By using ID-based encryption, they remove CRP requirement at a verifier side. It needs additional security association provider to guarantee AKE for a node and a verifier.

Recently, more lightweight PUF based authentication schemes [3], [18], [20] have been presented in IoT and RFID setting. First, in 2018, Gope and Sikdar have proposed a lightweight and privacy-preserving authentication scheme (GS) for IoT devices [19]. It utilizes a long-term secret and PUF as two-factor authenticators between an IoT device and a server. The protocol updates a one-time alias (identity) in a device every session, any outsider adversary cannot identify device at all. It preserves both security and privacy, but its authentication setting assumes a normal authentication scenario where a device and a server desire to authenticate each other with two-factor authenticators. In other words, it actually uses one same PUF between them as an authentication factor. In 2018, Gope et al. have also proposed a lightweight authentication protocol using PUF (GLQ) in a RFID setting. However, its RFID setting focuses on an authentication between a PUF-embedded tag and a reader with a shared same PUF, which is not end-to-end setting either. In 2018, one interesting authentication model has been suggested for software attestation in [3]. It assumes different types of PUF-embedded IoT devices. For attestation, each different device individually is verified by a system manager responsible for control, monitoring security. This authentication may consider different PUF-based devices, but note that it is still

in an authentication scenario between each PUF-embedded device and a verifier with a same PUF's output.

With regard to a general two-party setting, in 2011, a PUF-based KE was briefly introduced in a universal composition (UC) framework [8]. For dealing with PUF noise, a fuzzy extractor comprising a secret generation algorithm Gen was applied along with recovery algorithm Rep. In the enrollment phase, a server first evaluates a PUF(d = PUF(r)) for random value r and generates its secret s as $\{s, p\}$ = Gen(d). In the authentication phase, a client recovers s through Rep(PUF(r), p). Secret s is later served as a key to ensure confidentiality and integrity. This might be a typical PUF-based AKE, but its main contribution is to firstly model and design a UC-secure PUF-based AKE. Formal studies have been conducted on modeling a PUF in terms of realistic attack scenarios [15]. Such studies assumed a posterior access model in which an adversary achieves physical access to a PUF at least once. Further, a bad PUF model has been defined for a malicious adversary to cheat a malicious hardware that resembles a real PUF from outside [15]. In 2014, Rostami et al. have designed PUF-based AKE that are resilient against reverse-engineering attacks [36]. It is based on PUF's CRP between a prover and a verifier, but random subsets of the PUF response are sent to the verifier. The verifier authenticates the prover by matching the substring to full response string by an indexing secret key [36]. In 2018, a new attempt has been made by Zhang *et al.* [45]. They have proposed a transparent PUF-based user authentication with two-factor authenticators; PUF and voiceprint. It removes a typical interaction issue occurs between a user and a smart device and guarantees user-transparent two-factor authentication [45].

Some attempts have been made to combine a password with PUFs [5], [6], [17], [39] for efficient authentication. These studies used a password as an input of a PUF to generate a long-term secret key. First, Frikken *et al.* [17] presented a robust unilateral authentication (FBA) that only allows users with a PUF-based device with a memorable password to pass through the authentication step. The device is designed to output zero-knowledge proof of a discrete logarithm g^r. Subsequently, Resende *et al.* [39] build a PUF-based authentication scheme (RMA) by combining a PUF with a password-authenticated key exchange (PAKE) [10] between the user and server. In this scheme, a user first registers PUF-based secrets into DB and then recovers the secret at the authentication phase. By using the same secret, both parties finally go through the PAKE phase for creating a common session key. Very recently, Byun has built a PUF-based multi-factor AKE (MAKE) with a single PUF [5]. It has applied a single PUF into a well-known Bellare et. al.'s password-based AKE (PAKE) scheme [10]. Furthermore, Byun has designed a new MAKE (B19) in a generic way [6]. It first takes a secure PAKE and securely transforms it into a secure PUF-based MAKE scheme. It only supports a single PUF authentication setting where a user who has own PUF-embedded device desires to perform AKE with a server who previously keeps its corresponding PUF's unique output for multi-factor authenticators. Due to the generic design, the cost of communication and computation is relatively higher, but it can be easily and securely constructed if a secure and efficient PAKE is once given.

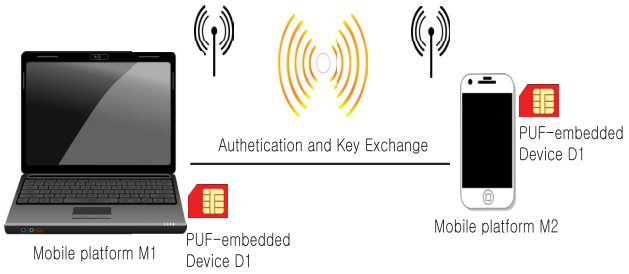

FIGURE 1. End-to-end authentication with different PUF-embedded devices.

B. OUR CONTRIBUTION

Our work achieves the same AKE goals but greatly differs from the previous works in the following aspects.

- *Distinct PUF-based mutual authentication between two parties:* The problem of AKE with a single PUF between a server and a client has been well researched in literature. In this paper, we consider a practical authentication scenario in which two users, each with a distinct PUF-embedded device, desire to establish an authenticated session key, as illustrated in Fig. 1. Our setting supposes that each user holds their own PUF-embedded device. For instance, a memory-based PUF, which exploits unstable characteristics of memory cells (SRAM, DRAM, and flip-flop) can be embedded in various devices such as PUF-embedded microSD, JAVA, and smart cards. These PUF-embedded devices are usually installed on a main platform (e.g., PC, cell phone, and smart electronic car) that is capable of communicating and computing for mutual authentication. In contrast, a PUF-based chip or device is originally soldered on and permanently detached from the mainboard of a platform. Then, each soldered PUF-based chip may output a secret information over the mainboard. Based on these outputs, the platforms finally exchange authenticated materials with other platforms for mutual authentication. A noticeable factor here is that both cases do not assume one PUF-based device between two parties but individual PUF-based unique devices belonging to each user. In this new setting, we first built a secure AKE with different PUFs.
- *Device oracle and PUF oracle:* In this study, we focus on a novel issue regarding a *missing device*. As a device is considered an equipment, it is usual to suppose that a PUF-embedded device can be stolen or lost. In addition, in case of permanently attached type of devices, they can be lost or stolen along with their platforms (PC, cell phone, etc.). In this case, an adversary is capable of asking queries to the device through the platform. To model this, we simply define a Device oracle

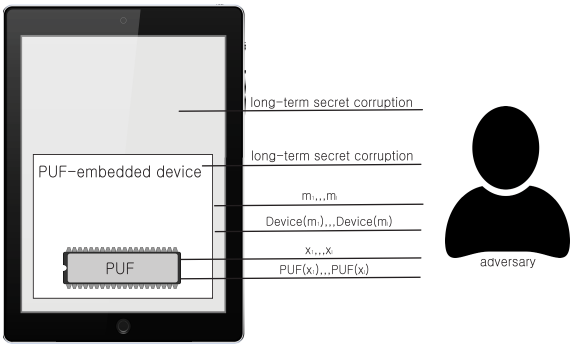

FIGURE 2. The ability of Adversaries.

and allow an adversary $\mathcal{A}$ to ask for inputs $m_1, \ldots m_l$ into a PUF-embedded device to obtain its outputs $o_1(=$ Devie$(m_1)), \ldots, o_l(=$ Devie$(m_l))$. In addition, we suppose that $\mathcal{A}$ can physically access a PUF at least more than once. This type of security on PUF has already been well studied in [8], [15]. Similarly, our security model also allows $\mathcal{A}$ to choose inputs $x_1, \ldots, x_l$ and achieve answers of $y_1, \ldots, y_l(=$ PUF$(x_1), \ldots,$ PUF$(x_l))$, except an answer for a target challenge. A corruption query for stealing long-term secrets, named Corrupt, is also newly considered in our model. Our model differs from the previous models in that two cases of Corrupt are defined: one for device corruption and the other for platform corruption. Fig. 2 illustrates the capabilities of $\mathcal{A}$. In conclusion, our construction is the first different PUF-based AKE that satisfies all security considerations under Device, PUF, and Corrupt oracles.

- *Operations in a device:* When we design operations inside PUF-embedded devices, we are required to consider whether their operations, steps, protocols should be designed in an identical manner. That is, in case of mass-produced PUF-embedded devices, protocols inside the device of $\mathcal{A}$ is usually equivalent to those on a device of $\mathcal{B}$. This is not only beneficial to a device company but also positively influences an open security policy or international standard. Through these identical operations on every device, it is possible for cryptographic designers to be able to expect the next outputs or consider the inputs that are appropriate for security. In addition, the designers can concentrate more on other important protocols carried over a platform. For these reasons, in our work, we assumed that every device has the same protocols (I-device). However, under I-device setting, designing a secure PUF-embedded AKE is more challenging; this is mainly because our model permits both Device and PUF oracles. That is, in our new model, an adversary can consistently achieve inputs/outputs for a device and PUF and then finally attempt to compute future or past session keys. We solved this issue in the new security model and built a secure AKE with different PUFs.
- *Efficiency:* As the previous protocols focus on a single PUF-based authentication system between a server and user, they consider a server's interventions not only to initially register PUF evaluations for users but also to authenticate for users. However, our distinct PUF-based AKE aims at an end-to-end authentication in a distributed network, and inherently its DB management cost can be reduced. It requires no intervention by a central server after an initial setup phase, and is hence suitable for practical decentralized networks. As analyzed in Table 1, our protocol requires only three flows to achieve an AKE with a distinct PUF, while other protocols need five flows or do not provide an common session key with mutual authentication.

III. COMPUTATIONAL ASSUMPTIONS AND CRYPTOGRAPHIC PRIMITIVES

A fuzzy extractor [16] is used to convert PUF's output into a uniform pseudorandom secret that serves as a secret key during authentication phase. For making a uniformly random session key, pseudorandom functions are applied in our protocol. They are defined as follows.

A. FUZZY EXTRACTORS

Definition 1 ([16]): An (M, m, l, t, ϵ)*-fuzzy extractor consists of two procedures,* Gen$(\cdot)$, Rep$(\cdot)$*;*

(1) Gen$(\cdot)$ *outputs a string* $R \in \{0,1\}^{l_r}$ *and a helper string* $P \in \{0,1\}^*$*, on input* W*, guaranteeing that for any distribution* W *with min-entropy* m*, if* $(R, P) \leftarrow$ Gen(W) *then* SD$((R, P), (U_l, P)) \leq \epsilon$*, where the* SD *means statistical distance.*

(2) Rep$(\cdot)$ *takes* (W', P) *on inputs and it outputs* Rep$(W', P) = R$ *if* dist$(W, W') \leq t$ *and* $(R, P) \leftarrow$ Gen(W).

B. PSEUDORANDOM FUNCTION

Definition 2: It is defined by a function $F_K : \mathbf{K}(F) \times \mathbf{D}(F) \rightarrow \mathbf{R}(F)$ *where* $\mathbf{K}(F) = \{0,1\}^k$ *is a set of keys and* $\mathbf{D}(F) = \{0,1\}^l$ *is a set of domain, and* $\mathbf{R}(F) = \{0,1\}^L$ *is a set of range of* F_K*. For any* PPT *algorithm* $\mathcal{B}$*, the followings are satisfied:*

(1) *Computation of* $\mathbf{F}_K(x)$ *should be efficient.*

(2) *The following prf-advantage,* ϵ_{prf} *is negligible.*

$$\epsilon_{prf} = |Pr[\mathcal{B}^{F_K} = 1 : K \xleftarrow{R} \mathbf{K}(F)] - Pr[B^F = 1 : F \xleftarrow{R} U_{F_K}]|$$

$\mathcal{B}^{F_K}$ *means that* $\mathcal{B}$ *only accesses* F_K *while* $\mathcal{B}^F$ *means that* $\mathcal{B}$ *only accesses* F *from* U_{F_K}*.* U_{F_K} *is a set of all functions from* $\mathbf{D}(F)$ *to* $\mathbf{R}(F)$.

C. PUF

We follow an ideal PUF model in which the PUF is considered as a random permutation oracle and an adversary adaptively query the PUF within polynomial time [8], [15], [17], [37], [39]. This *unpredictability* of PUF is formally defined in

TABLE 1. Comparison of relevant protocols.

SP[1]	SE[2]	Scheme	TFR[3]	TSC[4]	TSD[5]	TPE[6]	A[7]	K[8]	P[9]	F[10]	KW[11]
S	R	MMY [32]	3	$\approx 8 \cdot l_r$	$O(n)$	2	MA	N	Y	-	N
S	R	LNB [29]	3	$\approx 3 \cdot l_r$	$O(n)$	2	MA	N	Y	-	N
S	R	AC [1]	3	$\approx 6 \cdot l_r$	$O(n)$	2	MA	N	Y	-	N
S	R	KYWG [27]	4	$\approx 5 \cdot l_r$	$O(n)$	2	MA	N	N	-	N
S	R	XDLZW [45]	5	$\approx 9 \cdot l_r$	$O(n)$	2	MA	N	Y	-	N
S	R	GLQ [18]	3(4)	$\approx 7(8) \cdot l_r$	$O(n)$	1(2)	MA	N	Y	-	N
S	T	BFSK [8]	1	$\approx 6 \cdot l_r$	$O(n)$	$l+1$	UA	N	N	-	N
S	T	FBA [17]	2	$\approx 7 \cdot l_r$	$O(n)$	2	UA	N	N	-	Y[39]
S	T	RMA [39]	5	$\approx 8 \cdot l_r$	$O(n)$	$l+1$	MA	Y	N	Y	N
S	T	GS [19]	3	$\approx 8 \cdot l_r$	$O(n)$	2	MA	Y	Y	-	N
S	T	AS [3]	2	$\approx 8 \cdot l_r$	$O(n)$	l	UA	N	N	-	N
S	T	RMKWD [36]	3	$\approx 3 \cdot l_r$	$O(n)$	2	UA	Y	N	-	N
S	T	B19 [6]	3+3	$\approx 6+4 \cdot l_r$	$O(n)$	2	MA	Y	N	Y	N
D	T	Our scheme	3	$\approx 6 \cdot l_r$	-	2	MA	Y	N	Y	N

the second property of definition. The third property says that the PUF always responses output within bounded noise and the last property defines that the PUF should have uniqueness for each device. Our second property follows the definition in [37], and other two properties follows the definition in [17].

Definition 3 (PUF): A physically unclonable function $(t, \epsilon_1, \epsilon_2, \epsilon_{puf})$-PUF is a function $\{0,1\}^{l_p} \rightarrow \{0,1\}^{l_o}$ satisfying the followings.

(1) *A PUF is bound to a device and its evaluation is efficient.*

(2) *For any PPT adversary $\mathcal{A}_{puf}$, PUF's output is impossible to characteri1ze. Let PUF:$\{0,1\}^{l_p} \rightarrow \{0,1\}^{l_o}$ be an ideal PUF. An experiment $\mathbf{Exp}^{puf_b}_{\mathcal{A}_{puf}}(k)$ is defined for $b \xleftarrow{R} \{0,1\}$ where $k \in \mathbb{N}$ is a security parameter. $\mathbf{Exp}^{puf_0}_{\mathcal{A}_{puf}}(k)$ means an experiment in which $\mathcal{A}_{puf}$ can access an ideal PUF in polynomial number of times. That is, in this experiment, an oracle $\mathcal{O}^{\mathbf{PUF}}(\cdot)$ returns $y \leftarrow \mathbf{P}(x)$ on input $x \in \{0,1\}^{l_p}$. Otherwise if $b = 0$, in $\mathbf{Exp}^{puf_1}_{\mathcal{A}_{puf}}(k)$, $\mathcal{O}^{\mathbf{PUF}}(\cdot)$ returns a random $y \xleftarrow{R} \{0,1\}^{l_o}$ as an answer. $\mathcal{A}_{puf}$'s goal is to finally output a guessing bit b' for b. To be precise, the following puf-advantage, ϵ_{puf}, should be negligible.*

$$\epsilon_{puf} = |Pr[\mathbf{Exp}^{puf_1}_{\mathcal{A}_{puf}}(k) = 1] - Pr[\mathbf{Exp}^{puf_0}_{\mathcal{A}_{puf}}(k) = 1]|$$

(3) *The distance between two PUF's outputs on the same challenge x is at most t, e.g., the following probability is negligible ϵ_1.*

$$Pr[\mathbf{dist}(y,z) > t \mid x \leftarrow \{0,1\}^{l_p}, \\ y \leftarrow \mathbf{PUF}(x), \quad z \leftarrow \mathbf{PUF}(x)] \leq \epsilon_1$$

(4) *The distance between two outputs $\mathbf{PUF}_A(x)$, $\mathbf{PUF}_B(x)$ from different devices A, B, on the same challenge x, is at least t e.g., the following probability is negligible ϵ_2.*

$$Pr[\mathbf{dist}(y,z) < t \mid x \leftarrow \{0,1\}^{l_p}, \\ y \leftarrow \mathbf{PUF}_A(x), \quad z \leftarrow \mathbf{PUF}_B(x)] \leq \epsilon_2$$

D. COMPUTATIONAL DIFFIE-HELLMAN (CDH) ASSUMPTION

Definition 4 (CDH): We suppose that p is a strong prime order of a multiplicative group $\mathbb{G}_p$ and g is its generator. Let $\mathcal{A}_{cdh}$ be an adversary of CDH running in polynomial time. We define ϵ_{cdh} as probability that $\mathcal{A}_{cdh}$ succeeds in computing g^{xy} mod p from (g^x mod p, g^y mod p). The CDH assumption is that the probability, ϵ_{cdh} is negligible.

IV. SECURITY DEFINITION

In 2001, Bellare et at. have introduced a formal security model (for short, BPR) for a PAKE [10]. Since then, the model has been widely utilized for proving security of authentication and key exchange protocols for a single or multi-authentication factors (password, smartcard, fingerprint). Basically our security definitions are also based on the BPR. In addition, our definition considers a device-based authentication that allows for adversaries to access the target device as many times as they want. That is, our model defines Device query which models two kinds of behaviors that is able to access device or PUF. Since the PUF-embedded device contains PUF in a device, $\mathcal{A}$ should first get device then $\mathcal{A}$ can access PUF. In order to formalize this property, we define a new query Device($\Pi^i_A, \Omega_1, \Omega_2$). Inputs Ω_1, Ω_2 respectively mean that $\mathcal{A}$ can access device alone or device with PUF.

First let U be a set of identities for participants. Participants A (or B) may execute the protocol multiple times. We denote the i-th (resp. j-th) instance executed by entity A (resp. B) as Π^i_A (resp. Π^j_B) that models a participant A (resp. B) trying to authenticate B (resp. A) in instances i-th (resp. j-th). Let's consider permitted queries that model malicious behaviors for an adversary $\mathcal{A}$. The following queries are asked by $\mathcal{A}$.

- Send(Π^i_A, m), Send(Π^j_B, m). This query models that $\mathcal{A}$ can send an intended message m to Π^i_A (resp. Π^j_B). Then Π^i_A (resp. Π^j_B) outputs the next, honest message according to the protocol.

- **Reveal**(Π_A^i), **Reveal**(Π_B^j). If Π_A^i (resp. Π_B^j) accepts with a session key sk then this query outputs sk to $\mathcal{A}$.
- **Corrupt**(Π_A^i), **Corrupt**(Π_B^j). It outputs **A**'s (or **B**'s) long-term secrets stored in a platform. This query is modeled for a corruption of long-term key, not ephemeral values in a platform.
- **CorruptD**(Π_A^i), **CorruptD**(Π_B^j). It outputs **A**'s (resp. **B**'s) long-term secrets stored in a **PUF**-embedded device regardless of **PUF**. Its scope is limited to a long-term secret (not ephemeral value) in a **PUF**-embedded device as in **Corrupt** query.
- **Device**($\Pi_A^i, \Omega_1, \Omega_2$), **Device**($\Pi_B^i, \Omega_1, \Omega_2$). If Ω_1 is not empty and Ω_2 is empty then this query models that $\mathcal{A}$ can access **A**'s (resp. **B**'s) device with an input Ω_1. That is, **A**'s (resp. **B**'s) device takes an input Ω_1 and processes Ω_1 by inside instructions. The device finally returns its output as an answer. If Ω_1 is empty and Ω_2 is not empty then it models that $\mathcal{A}$ wants directly access PUF_a (resp. PUF_b) in a **PUF**-embedded device with an input Ω_2, and finally $\mathsf{PUF}_a(\Omega_2)$ (resp. $\mathsf{PUF}_b(\Omega_2)$) is returned as an output. The notion PUF_a (resp. PUF_b) means **A**'s **PUF** (resp. **B**'s **PUF**) which is embedded in **A**'s (resp. **B**'s) device. If neither Ω_1 nor Ω_2 are empty cases then $\mathcal{A}$ is able to access both **Device**($\Pi_A^i, \Omega_1, \cdot$) and **Device**($\Pi_A^i, \cdot, \Omega_2$) queries. The notion $\cdot$ means empty value.
- **Execute**(Π_A^i, Π_B^j). It returns protocol messages that comes through an honest execution between Π_A^i and Π_B^j.
- **Test**(Π_A^i). This query is used to measure $\mathcal{A}$'s advantage on a target key. $\mathcal{A}$ should distinguish the target key is a real session key or not. That is, when this query is asked, a coin is flipped to determine b. If $b = 1$, a real session key is returned to $\mathcal{A}$. Otherwise, a random key is randomly selected and returned to $\mathcal{A}$.

The notion of partnering is used as it is defined in [10] while the notion of freshness is lightly modified in terms of corruption queries.

Definition 5 (Session Identifier, Partner id, and Partnering [10]): We denote $\mathbf{sid}_A^i$ *a session identifier for* Π_A^i. *It is a concatenation of protocol messages sent and received by* Π_A^i. *A partner ID is defined as a corresponding participant with whom* Π_A^i *is interacting, which is denoted as* $\mathbf{pid}_A^i$. *Instances* Π_A^i *and* Π_B^j *are partnering if the following conditions are satisfied: (1) Both instances have accepted with a same session key, session ID, partner ID; (2) no oracle other than* Π_A^i *and* Π_B^j *accept with the pid of* Π_A^i *and* Π_B^j ($\mathbf{pid}_A^i = \Pi_B^j$, $\mathbf{pid}_B^j = \Pi_A^i$).

Definition 6 (Freshness and Session Key Security): Suppose that $\mathcal{A}$ *asks a* **Test**(Π_A^i) *query. A session key is said to be fresh if they meet the following conditions: (1) Neither* **Reveal**(Π_A^i) *nor* **Reveal**(Π_B^j) *queries are asked by* $\mathcal{A}$ *where* Π_A^i *are partnering with* Π_B^j; *(2) Regarding corrupt queries, trivial cases should be excluded. That is, if* $\mathcal{A}$ *is able to access long-term secrets (***Corrupt**(Π_A^i)) *and* **PUF***-embedded device (***Device**($\Pi_A^i, \Omega_1, \Omega_2$)) *together, then it is easy for* $\mathcal{A}$ *to distinguish a target session key. Thus, the following formula should be false. (Here, it is defined that if* $\mathcal{A}$ *has asked a query of* **Corrupt**(Π_A^i) *then* **C**(Π_A^i) = **true**, *otherwise it is* **false**. *The same is applied for* **C**(Π_B^j), **D**($\Pi_A^i, \Omega_1, \Omega_2$), **D**($\Pi_B^j, \Omega_1, \Omega_2$).*)*

$$(\mathbf{C}(\Pi_A^i) \wedge \mathbf{D}(\Pi_A^i, \Omega_1, \Omega_2)) \vee (\mathbf{C}(\Pi_B^j) \wedge \mathbf{D}(\Pi_B^j, \Omega_1, \Omega_2)) = \mathbf{false}$$

For a **Test** *query, a coin bit b is determined.* $\mathcal{A}$ *outputs a guessing bit* b' *for b.* $\mathcal{A}$*'s advantage is defined as* $\mathbf{Adv}_P^{sk}(T, k) = 2\,Pr[b = b'] - 1$. *A given protocol P is session key secure if* $\mathbf{Adv}_P^{sk}(T, k)$ *is bound to a negligible function* $\varepsilon(k)$ *in k for probabilistic polynomial time T adversary* $\mathcal{A}$,

$$\mathbf{Adv}_P^{sk}(T, k) \leq \varepsilon(k),$$

where k is a security parameter.

V. AKE WITH DIFFERENT PUF-EMBEDDED DEVICE

A. THE FIRST PROTOCOL

The protocol is made up of setup and authentication phases.

1) SETUP PHASE

In our setting, we suppose two authentication factors; (1) two users (**A** and **B**) share a common secret **S** and (2) they carry each own devices embedded with **PUF**. We denote each **PUF**-embedded device by PUF_b, PUF_a, respectively. Through two factors, two users prepare initial value for secure authentication. As illustrated in Fig. 3, a user **A** obtains w_a by evaluating $\mathsf{PUF}_a(H_1(\mathsf{S}))$ then also obtains (r_a, p_a) from computation of $\mathsf{Gen}(w_a)$. In the same way, **B** obtains (r_b, p_b). Finally users **A** and **B** securely exchange r_a, r_b and maintain (S, r_b, p_a), (S, r_a, p_b) as long-term secrets, respectively. H_1 is a cryptographic secure hash function $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$.

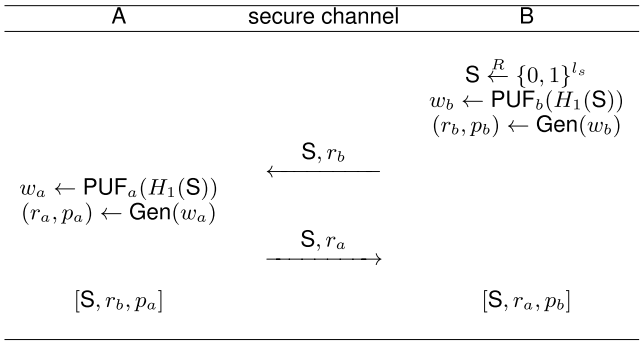

FIGURE 3. Setup phase.

2) AKE PHASE

In Fig. 4, we illustrate a sequence of operations performed in each device into square boxes with its inputs and outputs.

(1) A user **A** selects a random value N_a and sends it to **B**
(2) A user **B** sets $m_b = (N_a||A||B)$ and inputs (S, m_b) into PUF_b. Upon receiving (S, m_b), PUF_b obtains $\tilde{w_b}$ by evaluating $\mathsf{PUF}_b(H_1(\mathsf{S}))$ and computes r_b from $\mathsf{Rep}(\tilde{w_b}, p_b)$.

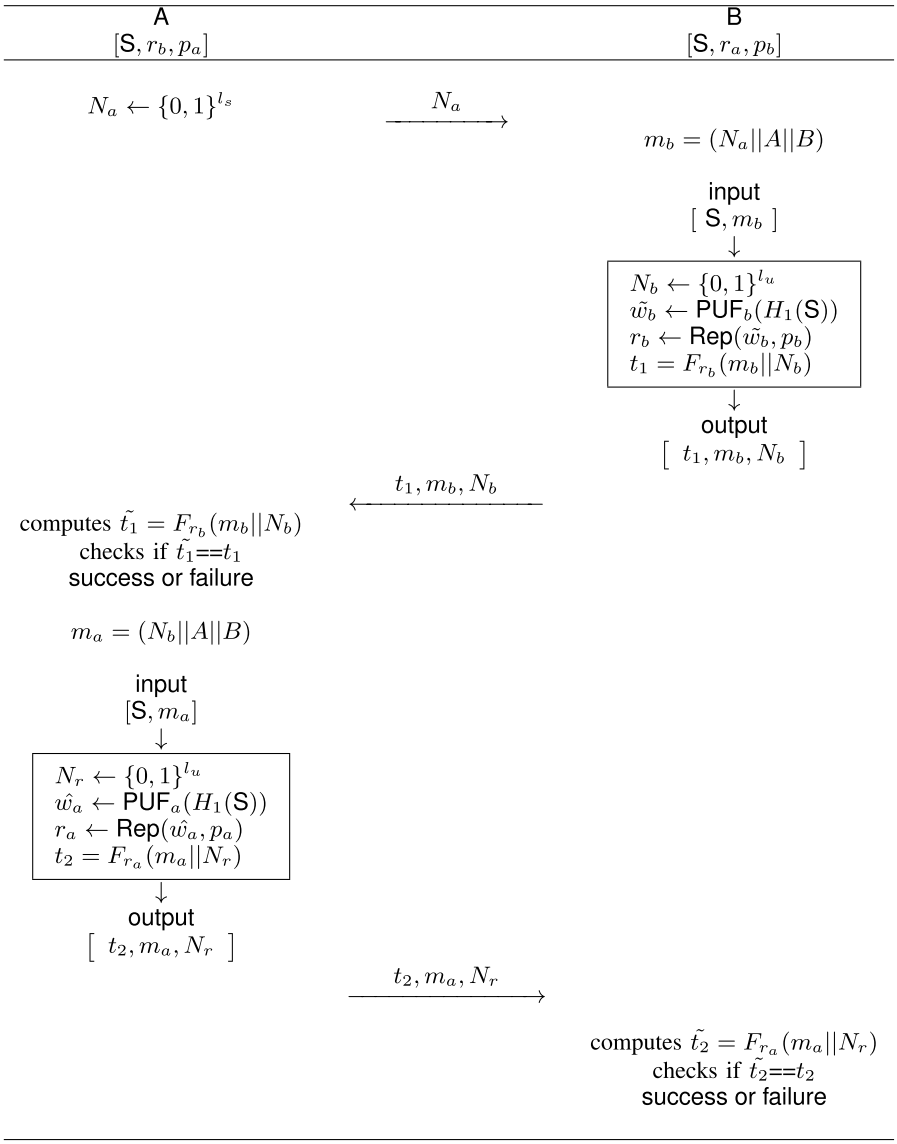

FIGURE 4. AKE phase.

Using the key r_b, an authentication tag t_1 is computed from $F_{r_b}(m_b||N_b)$. PUF_b finally outputs (t_1, m_b, N_b) where $t_1 = F_{r_b}(m_b||N_b), m_b = N_a||A||B$. B directly sends them to A.

(3) Since a user A holds S and r_b, A can compute $\tilde{t_1} = F_{r_b}(m_b||N_b)$ and check if $\tilde{t_1}$ is equal to t_1. If it is equal, B's authentication is successful, otherwise, it is a failure. In the same way, A sets $m_a = (N_b||A||B)$ and puts (S, m_a) into PUF_a and obtains the output (t_2, m_a, N_r). A sends (t_2, m_a, N_r) to B. Then, B can compute $\tilde{t_2} = F_{r_a}(m_a||N_r)$ and check if it is equal to the received t_2. If the check is valid then A's authentication is successful, otherwise, it is a failure.

3) WEAKNESS ANALYSIS

The first protocol has two weaknesses, as follows.

- Let's consider that an adversary $\mathcal{A}$ only knows a secret S. In the protocol, if the secret S is revealed to $\mathcal{A}$, the authentication tag t_1 cannot be produced without B's device. Moreover, $\mathcal{A}$ cannot impersonate A or B with only the secret S because the authentication protocol necessarily uses r_a or r_b for making tags t_2, t_1. However, what if $\mathcal{A}$ gets to obtain long-term keys such as r_a or r_b then? Unfortunately, if $\mathcal{A}$ has r_a or r_b then $\mathcal{A}$ is able to impersonate B or A without devices. For example, $\mathcal{A}$ can compute $t_2 = F_{r_a}(m_a||N_r)$ without any PUF evaluations in A's devices if once $\mathcal{A}$ steals r_a. It renders A's device useless in terms of an authentication factor. Our goal is to design an authentication protocol that only valid user who has a valid device and long-term secrets should be able to authenticate mutually. In the symmetric setting, the corruption of A's long term secret means to be able

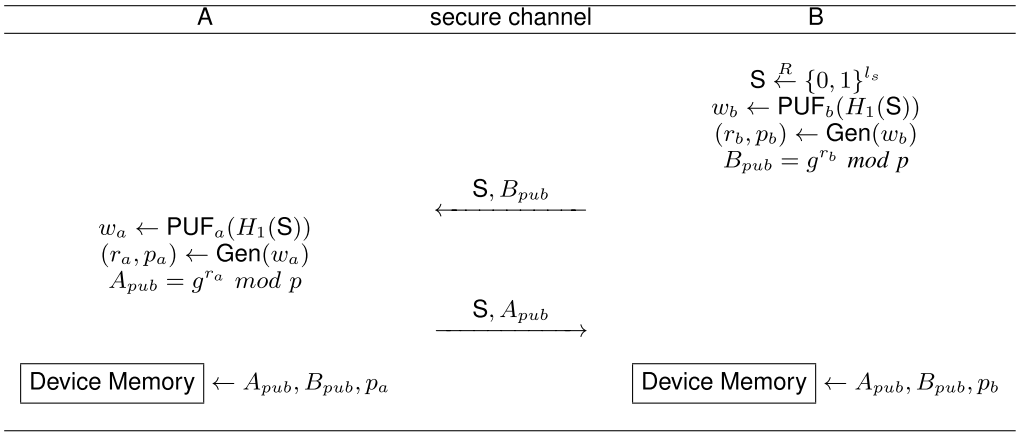

FIGURE 5. Setup phase.

to impersonate A. However, the problem here is, through corruption A, $\mathcal{A}$ can impersonate B, and vice versa.

- Second, the protocol only focuses on how to mutually authenticate based on authentication factors. That is, it hasn't addressed on how to securely agree an authenticated session key. In addition, the initial setting requires three long-term distinct secrets for each user ($[\mathsf{S}, r_a, p_b]$), which might be one of big burdens for users. One solution is to keep those secrets in the memory of device, if it is possible.

We next present an enhanced version that overcomes these all weaknesses.

B. THE SECOND PROTOCOL

1) SETUP PHASE

The same authentication setting is considered as in the first protocol. Before authentication, users A and B compute (r_a, p_a) and (r_b, p_b) by using their own $\mathsf{PUF}_a(\cdot)$ and $\mathsf{PUF}_b(\cdot)$ functions with an input of a common secret S. As shown in Fig. 5, they securely exchange and keep (A_{pub}, B_{pub}, p_a) and (A_{pub}, B_{pub}, p_b), respectively, in each device, where A_{pub}, B_{pub} are defined by $g^{r_a} \bmod p$, $g^{r_b} \bmod p$. Then the secret S is securely maintained in a user-side platform while (A_{pub}, B_{pub}, p_a) (or (A_{pub}, B_{pub}, p_b)) is securely stored in A's (or B's) device. The public parameter p is a strong prime order of a multiplicative group $\mathbb{G}_p$ and g is its generator. H_1, H_2 are hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_{h_1}}$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_{h_2}}$ where H_1 is a random hash orale and H_2 is a collision resistant hash function that is infeasible to find a collision $x \neq y$ satisfying $H_2(x) = H_2(y)$. When making $A_{pub}(= g^{r_a} \bmod p)$ (or $B_{pub}(= g^{r_b} \bmod p)$), a group hash function (or programmable hash function [21]) that efficiently maps an arbitrary string r_a (or r_b) into a group $\mathbb{G}_p$ where a discrete logarithm problem is hard regarding r_a(or r_b) is applied in our protocol. Here, for simplicity, we omit its notation.

2) AKE PHASE

(1) First a user A selects a random value N_a and sends it with an identifier A to B.

(2) On receiving N_a, A, B sets $m_b = (N_a||A||B)$. Then B puts $[i_1(=\mathsf{S}), i_2(=m_b), i_3(=\emptyset)]$ into B's device. Then, it obtains $(\tilde{w_b}, r_b)$ from executions $\tilde{w_b} = \mathsf{PUF}_b(H_1(\mathsf{S}))$ and $r_b = \mathsf{Rep}(\tilde{w_b}, p_b)$. A random number N_b is chosen and m_B is newly set by $m_b||N_b$ inside a device. B's device computes $v = F_d(m_b||B_{pub})$ where $d = H_2((A_{pub})^{r_b} \bmod p)$. If v is equal to i_3 (or i_3 is empty value) then B's device computes $t_b = F_d(m_B||A_{pub}), k_1 = F_d(m_B||1), k_2 = F_d(m_b||1)$ where $d = H_2((A_{pub})^{r_b} \bmod p)$, and finally outputs (m_B, t_b, k_1, k_2). The messages (m_B, t_b) is only sent to A for B's authentication. A session key material k_B is initialized by k_1 and later used for making a real session key.

(3) In order to verify (m_B, t_b), A needs to compute d first. A inputs $[i_1(=\mathsf{S}), i_2(=m_B), i_3(=t_b)]$ into own device. Internally, with regard to the identical S, A's device can recover the identical r_a from computing $\mathsf{Rep}(\tilde{w_a}, p_a)$ where $\tilde{w_a} = \mathsf{PUF}_a(H_1(\mathsf{S}))$. A random number $\tilde{N_a}$ is chosen and m_A is initialized by $m_B||\tilde{N_a}$. From calculation of $H_2((B_{pub})^{r_a} \bmod p)$, it computes $v = F_d(m_B||A_{pub})$ and checks if $i_3(= t_b)$ is equal to v. If it is valid then A's device computes $t_a = F_d(m_A||B_{pub}), k_1 = F_d(m_A||1), k_2 = F_d(m_B||1)$ and concludes B's authentication is successful. Otherwise, its authentication is failed. Furthermore, for A's authentication, (m_A, t_a) is delivered to B. In the same way, B inputs $[i_1(=\mathsf{S}), i_2(=m_A), i_3(=t_a)]$ into B's device and then it internally computes r_b through $\widehat{w_b} = \mathsf{PUF}_b(H_1(\mathsf{S}))$, $\mathsf{Rep}(\widehat{w_b}, p_b)$. It computes the same $d = H_2((A_{pub})^{r_b} \bmod p)$ and computes $v = F_d(m_A||B_{pub})$. A random number N'_b is chosen and m'_B is newly set by $m_B||N'_b$. If t_a is equal to v then A's authentication is successful. Otherwise, it outputs an authentication failure. It finally computes

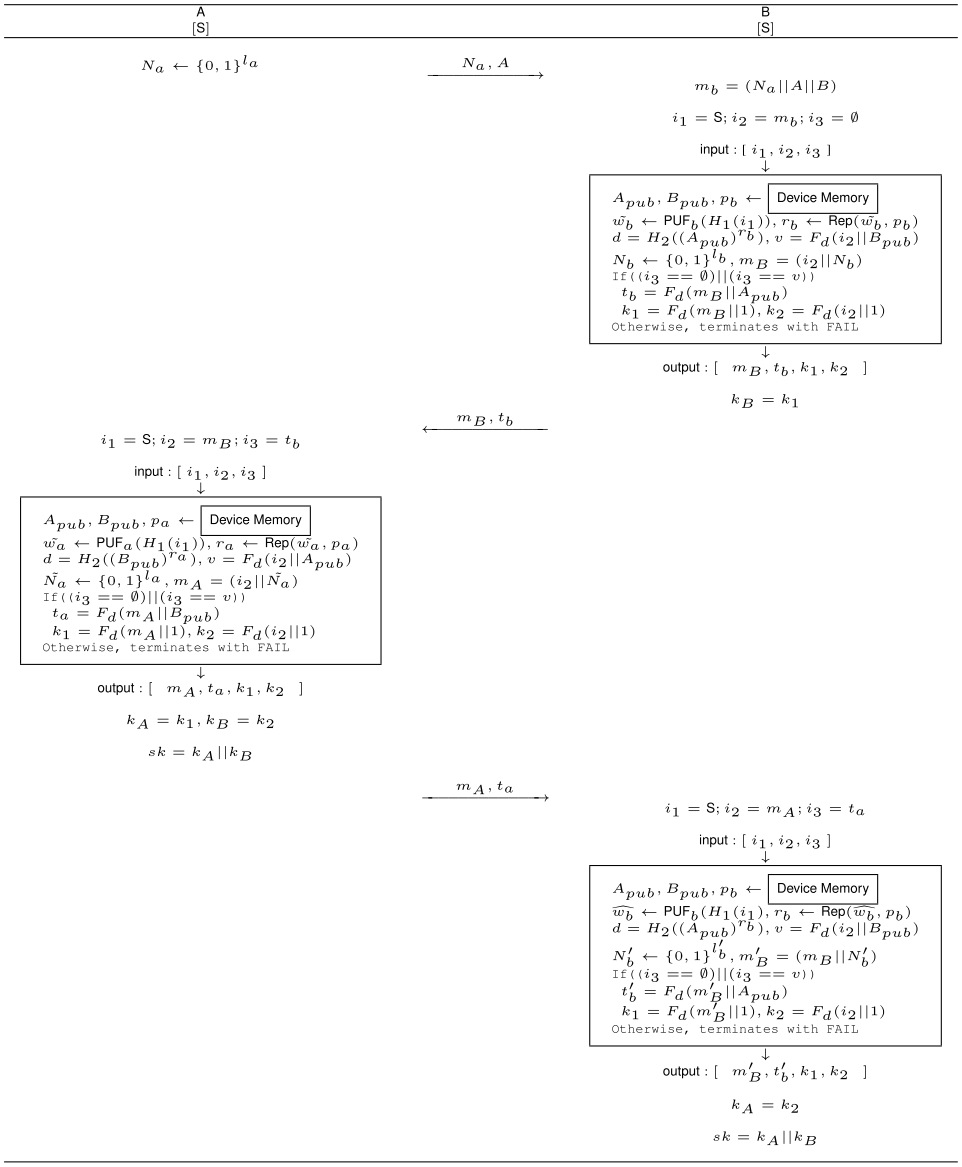

FIGURE 6. AKE phase.

$t'_b = F_d(m'_B||A_{pub}), k_1 = F_d(m'_B||1), k_2 = F_d(m_A||1)$. B then sets k_A be k_2 and makes a session key $sk = k_A||k_B$.

VI. PROTOCOL DISCUSSION

A. DEVICE OPERATION

First, device operations should be designed in a consistent manner with identical steps and operations until it outputs a valid answer. For mutual authentication, at least three accesses to devices are required; one access for a sender (initiator) and two accesses for a receiver, as shown in Fig. 6. Our device has been designed to take three inputs i_1, i_2, and i_3. Input i_1 is used for long-term secret S and input i_3 is used for an authentication tag for input i_2. One important principle is that each device is able to produce authentication tag i_3 for i_2 with the help of input $i_1 (= \mathsf{S})$. Thus, the verification task for the authentication tag is performed in advance in each device. However, an exceptional case exists, in which receiver B first receiving challenge message (N_a, A) does not comprise authentication tag i_3 derived from PUF_a. Even though our protocol is designed for A to first use its device, it cannot initially have an authentication tag. For handling such a case, we additionally designed a null($\emptyset$) condition by using the *if* statement inside a device so that the remaining operations may be continuously executed over the device.

B. SHOULD RANDOM NUMBER BE GENERATED IN THE DEVICE?

The generation of random numbers inside a device requires operational costs. However, for security reasons, random-number generation is unavoidable. If we assume that random

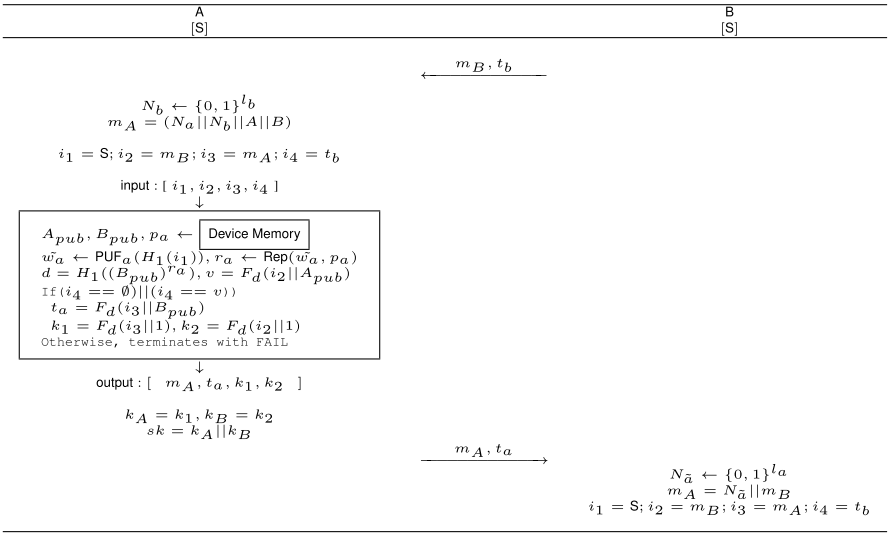

FIGURE 7. AKE phase.

numbers are generated outside the device, for example at the platform level of B, as illustrated in Fig. 7, this may result in a vulnerability against forward secrecy. The protocol in Fig. 7 is not a full protocol but only illustrates certain part of the protocol of A. To take a random input from the outer platform, a device of A may be designed to achieve four inputs and outputs instead of generating random numbers. Thus, as in Fig. 7, the newly generated value (m_A) by A and the received values from B (m_B, t_b) are used as inputs for A's device. Normally, forward secrecy guarantees that $\mathcal{A}$ cannot create the past session key after an adversary $\mathcal{A}$ raises $\mathsf{Device}(\Pi_A^i, \Omega_1, \Omega)$ and $\mathsf{Corrupt}(\Pi_A^i)$ queries. Let us suppose that $\mathcal{A}$ captures a valid message m_B, t_b, m_A, t_a from honest executions and accesses $\mathsf{Device}(\Pi_A^i, \mathsf{S}||m_B||m_A||t_b, \cdot)$ oracle where S is an answer of the $\mathsf{Corrupt}(\Pi_A^i)$ query. Then, it outputs $[m_A, t_a = F_d(m_A||B_{pub}), k_1 = F_d(m_A||1), k_2 = F_d(m_B||1)]$ as an answer. Values m_A, t_a are sent to A but $k_1 = F_d(m_A||1), k_2 = F_d(m_B||1)$ are directly used for creating a session key. This clearly violates a *forward secrecy* that ensures secrecy for a past session key ($sk = k_1||k_2$) against future compromises of long-term secrets. Therefore, for security reason, we designed for each device to create random numbers by itself. This does not imply that an approach of generating random numbers inside a device is the only way to achieve forward secrecy. There might be other ways, but such a method has not been found yet. This might be an interesting future work.

C. FORWARD SECRECY

As shown in Fig. 6, our device deals with the generation of a random value. Let us suppose a same situation in which $\mathcal{A}$ asks $\mathsf{Device}(\Pi_A^i, \mathsf{S}||m_B||t_b, \cdot)$ for secret S, in which m_B, t_b are values collected from past honest executions. Our protocol is designed for the device to create a new key material $k_1 = F_d(m_A||1), k_2 = F_d(m_B||1)$ for a new random value $m_A = m_B||\tilde{N_a}$ whenever a session key is generated. Hence, answer k_1 can no longer be the key material for the past session key. Further, $\mathcal{A}$ may use the device of B with secret S. That is, $\mathcal{A}$ captures m_A, t_a, m_B, t_b from the past honest executions and passes a query to $\mathsf{Device}(\Pi_B^j, \mathsf{S}||m_A||t_a, \cdot)$. The Device answers with $m'_B = m_b||N_b = N_a||A||B||N_b, t'_b = F_d(m'_B||A_{pub}), k_1 = F_d(m'_B||1), k_2 = F_d(m_A||1)$. Among these, $k_2(= F_d(m_A||1))$ is the important keying material for the past session key. However, for creating a complete session key, $F_d(m_B||1)$ for m_B is also required. Thus, $\mathcal{A}$ may put forward a query to $\mathsf{Device}(\Pi_B^j, \mathsf{S}||m_B||t_b, \cdot)$, for which the answer would always be FAIL. This is because $\mathsf{Device}(\Pi_B^j, \mathsf{S}||m_B||t_b, \cdot)$ first checks if $t_b(= F_d(m_B||A_{pub}))$ is equal to $v = F_d(m_B||B_{pub})$, and then outputs FAIL. The critical point is that $\mathcal{A}$ would never be able to obtain tag value $v = F_d(m_B||B_{pub})$ from any Device oracle as v is generated using B_{pub} (not A_{pub}) and t_b is generated using A_{pub} inside the device. Hence, $\mathcal{A}$ cannot manipulate v. What if $\mathcal{A}$ could access two Device oracles by using secret S? In this case, through the queries $\mathsf{Device}(\Pi_A^i, \mathsf{S}||m_B||t_b, \cdot)$ and $\mathsf{Device}(\Pi_B^j, \mathsf{S}||m_A||t_a, \cdot)$, $\mathcal{A}$ can obtain each valid key material $F_d(m_B||1), F_d(m_A||1)$ without failures. In some ways, it might be a usual consequence in the symmetric setting that if $\mathcal{A}$ with secret S could access two users' devices, then $\mathcal{A}$ could surely obtain past or future session keys. This remains a good future work to guarantee forward secrecy under two device accesses with secret S. We further discuss this issue with an asymmetric setting in Section VII.

D. IMPERSONATION ATTACK WITH DEVICE ORACLE

In an AKE phase, B first uses its device with input $(\mathsf{S}, m_b, \emptyset)$. Basically, input i_3 serves as an authenticator received from A. In B, there cannot exist an authenticator from a device of A because A is the first sender and B is the first receiver. However, one security issue here is that $\mathcal{A}$ may intentionally

abuse $\emptyset$ to paralyze the authentication validation process in the device. That is, in our device design, $\mathsf{Device}(\Pi_A^i, i_1||i_2||\emptyset, \cdot)$ always outputs answers without FAIL, regardless of i_1 and i_2. First, if we suppose that $\mathcal{A}$ possesses secret S from the Corrupt query and accesses two Device oracles, then $\mathcal{A}$ can obviously impersonate A and B. However, the issue in such a situation is as follows: "*Is $\mathcal{A}$ able to impersonate A (or B) with only S or Device oracles? (only with one authenticator factor)*. First, suppose that $\mathcal{A}$ uses $\mathsf{Device}(\Pi_A^i, \mathsf{S}'||M||\emptyset, \cdot)$ for an arbitrary S'. Then, its answer is $[M||r, f_{d'}(M||r||0), f_{d'}(M||r||1), f_{d'}(M||r||1)]$, where $d' = H_1((A_{pub})^{r_b'})$, $r_b' = \mathsf{Rep}(\mathsf{PUF}_b(H_1(\mathsf{S}')))$ and r is a random value chosen inside the device. However, those outputs are useless owing to invalid value d'. In contrast, if $\mathcal{A}$ only knows S, it still cannot obtain valid d without accessing the Device oracle. Therefore, in our scheme, to validly use the Device oracle, valid secret S is necessary. Hence, only valid users who hold S and the device are able to authenticate each other with a common session key.

E. IS THE DEVICE MEMORY NECESSARY?

First, suppose that $\mathcal{A}$ obtains secrets (A_{pub}, B_{pub}, p_b) in the device memory (DM) of B. For producing valid messages, m_B, t_b, k_1, k_2, the most important step is to calculate d. Although $\mathcal{A}$ obtains A_{pub}, B_{pub}, p_b, it encounters another challenge on the calculation of d to forge authenticator t_b. That is, without d, $\mathcal{A}$ can never fake authenticators. Moreover, in our scheme, computations of d require valid secret S and a device. However, if we design A_{pub}, B_{pub}, and p_b on a platform (outside the devices), then we must design new input i_4 for handling A_{pub}. In this case, for instance, $\mathcal{A}$ may have A_{pub} from the Corrupt query and can produce authenticator t_a (or t_b) for i_2 through the Device oracles; this violates forward secrecy, as discussed earlier. That is, from the past valid transcripts $(m_B, t_b = F_d(m_B||A_{pub}), m_A, t_a = F_d(m_A||B_{pub}))$, $\mathcal{A}$ can ask query $\mathsf{Device}(\Pi_A^i, \mathsf{S}||m_B||t_b||A_{pub}, \cdot)$ and $\mathsf{Device}(\Pi_A^i, \mathsf{S}||m_A||t_a||B_{pub}, \cdot)$, and successfully obtain answers $F_d(m_B||1)$ and $F_d(m_A||1)$ without needing d, which are the main keying materials for a session key. Thus, in conclusion, A_{pub}, B_{pub} should be processed in A's, B's device, respectively (and not be an input outside) with secret d. Therefore, we designed a DM in the device.

F. ANONYMOUS IDENTIFIER

One possible argue may be that our protocol does not provide privacy while the recent schemes [18], [19] guarantee privacy in an anonymous ID aspect. First, in their network model, there are servers aiming to authenticate a PUF-embedded device. The server is able to regularly update a temporary identifier (TID) over a database. By changing TID every session, they preserve anonymous identifier for devices. However, our setting aims a distributed setting without a central server, hence it is hard to maintain a database for keeping many user's TID. To protect identifier, an encryption approach may be one of candidates in our setting. For instance, to hide identifiers A, B, our protocol may send $m_B(= N_a||A||B||N_b)$, $m_A(= N_a||A||B||N_b||\tilde{N_a})$ in an encrypted form. However, for making encryption, new encryption/decryption keys should be established beforehand. Besides, those keys should be securely maintained from outsider adversaries. Then, once if Corrupt query is asked, all identifiers are revealed to adversaries. Moreover, our security model allows a new Device and PUF oracles for PUF-to-PUF authentication. That is, any adversary can access any user's Device and PUF on behalf of the participants. Hence it is impossible to perfectly satisfy identifier protection under such Device and PUF query model. For these reasons, in this paper, an issue of anonymous identifier has not been addressed. It remains a good future work on how we handle and define a privacy model for PUF-to-PUF setting.

G. APPLICATION FOR MULTI-USER SETTING

A multi-user setting assumes multi-pair of end users. It is different from a group-based AKE where group members desire to share a group key. Our scheme has been designed with PUF-based device that is embedded into IoT platform such as laptop or tablet, as illustrated in Fig. 1. In view of implementation, our network setting can be a decent architecture when we extend our scheme into a multi-user IoT settings. When our scheme is implemented with two different types of PUF-embedded device, two IoT devices communicate each other through each own TCP/IP socket. After socket is established, our scheme does execute AKE phase for making a common session key without any involvement of trust parties. However, our AKE scheme has a structural limitation. Since our AKE is based on symmetric cryptographic primitives, each device cannot do authentication phase without a setup phase. That is, if we suppose n PUF-embedded devices, then $\frac{n(n-1)}{2}$ setup phases should be prepared before actual AKE phase starts. It also does mean that each user should previously know all participants they desire to communicate.

H. STATIC SCENARIO IN A SYMMETRIC SETTING

Our authentication model only supports a static scenario in which designated users are only considered beforehand. All users with embedded PUF devices should do setup phase before deployment. After deployment once, users cannot join with a new user without a registration phase. It does not mean that our model does not accept new users at all. That is, even after deployment, if a new user starts the setup phase once with a target user, afterwards they can do AKE. But, during the setup phase, involved users cannot do anything (or AKE) but do exchange initial data in a secure way for the setup phases. This is a natural consequence for sharing secrets in a symmetric setting. In our scheme, each PUF-embedded device requires one PUF evaluation with one round communication during a setup phase.

I. SECURITY PROOF

We define sequences of games and the following well-known lemma is used in the security proof.

Lemma 5.1 [40] Let E, E' and F be events defined on a probability space such that $\Pr[E \wedge \neg F] = \Pr[E' \wedge \neg F]$. Then, we have $|\Pr[E] - \Pr[E']| \leq \Pr[F]$.

Theorem 5.1 Let $\mathcal{A}$ be a polynomial time adversary against AKE protocol P within a polynomial time bound T. $\mathcal{A}$ can ask q_s send queries, q_e execute, q_{h_1} hash, q_d device queries. Then,

$$\mathsf{Adv}_P^{sk}(T', k) \leq \frac{(q_e + q_s)^6}{2^{3l_r+2}} + \frac{q_{h_1}^2}{2^{l_{h_1}}} + 2\epsilon_{puf} + \frac{2q_d^4(2^{2L} + 2^{l_s})}{2^{2L+l_s}} + 2\epsilon_{cdh}$$

where p is a strong prime order of a group $\mathbb{G}_p$ and T' is a polynomial bound time.

Proof: We define several games from the real game $\mathbf{G}_0$ to $\mathbf{G}_4$. In each game, when $\mathcal{A}$ asks a Test query, a coin for bit d is flipped to specify the answer on a real key or a random key. Then $\mathcal{A}$ guesses a bit d and outputs d'. Let S_i be an event that $d' = d$ in $\mathbf{G}_i$ and $\Pr[\mathsf{S}_i]$ is its probability.

Game G_0: The setup phase is assumed to be performed in a secure channel. Then an authentication protocol P is executed against an adversary $\mathcal{A}$. The session key advantage of $\mathcal{A}$ is defined as

$$\mathsf{Adv}_P^{sk}(T, k) = 2\Pr[\mathsf{S}_0] - 1 \quad (1)$$

Game G_1: In this game, Device, Corrupt, CorruptD, Send, Reveal, and Execute queries are simulated as follows.

- H_1 query: For simulation H_1, we maintain a list table $\mathcal{L}_{H_1}$. When $\mathcal{A}$ asks a hash query of m to H_1 such that a record (m, r) appears in $\mathcal{L}_{H_1}$, then r is outputted as an answer of $H_1(m)$. Otherwise, a new $r \in \{0,1\}^{l_{h_1}}$ is chosen at random and returned, then the record (m, r) is added to the list $\mathcal{L}_{H_1}$.
- $\mathsf{Device}(\Pi_A^i, \cdot, m)$, $\mathsf{Device}(\Pi_A^i, \cdot, m)$ query: $\mathbf{G}_1$ queries PUF_a (resp. PUF_b) with m and obtains $y = \mathsf{PUF}_a(m)$ (resp. $\mathsf{PUF}_b(m)$), which is returned as an answer.
- $\mathsf{Device}(\Pi_A^i, m, \cdot)$ query: Basically the answer is made according to the protocol. The input m is parsed into m_1, m_2, m_3. The game $\mathbf{G}_1$ first asks H_1 oracle with m_1 and gets $r = H_1(m_1)$ through H_1 defined above. Then $\mathbf{G}_1$ queries PUF_a with r and obtains an answer $\mathsf{PUF}_a(r)$ and then finally computes $r_b = \mathsf{Rep}(\mathsf{PUF}_a(r), p_a)$. Then $\mathbf{G}_1$ computes $v = F_d(m_2||A_{pub})$ where $d = H_2((B_{pub})^{r_a} \bmod p)$. It chooses a random number $\tilde{N}_a$ and $m_A = m_2||\tilde{N}_a$. If $v = m_3$ then it makes $t_a = F_d(m_A||B_{pub}), k_1 = F_d(m_A||1), k_2 = F_d(m_2||1)$ for a random number $\tilde{N}_a$. It finally returns m_A, t_a, k_1, k_2 as an answer.
- $\mathsf{Device}(\Pi_B^j, m, \cdot)$ query: The process is same with the previous rule except notations. $\mathbf{G}_1$ first parses m into m_1, m_2, m_3. It asks H_1 oracle with an input m_1 and obtains $r = H_1(m_1)$. Then $\mathbf{G}_1$ queries PUF_b with r and obtains an answer $\mathsf{PUF}_b(r)$ and computes $r_b = \mathsf{Rep}(\mathsf{PUF}_b(r), p_b)$ and $d = H_2((A_{pub})^{r_b} \bmod p)$. $\mathbf{G}_1$ chooses a random value N_b and sets $m_B = m_2||N_b$. Then $\mathbf{G}_1$ is able to check if v is equal to $F_d(m_2||B_{pub})$. If they are equal then it computes $t_b' = F_d(m_B||A_{pub}), k_1 = F_d(m_B||1), k_2 = F_d(m_2||1)$ for a random number N_b. It finally returns m_B, t_b', k_1, k_2 as an answer.
- Send queries are simulated as follows.
  - $\mathsf{Send}(\Pi_A^i, start)$: A random value N_a and an identifier A are outputted as an answer.
  - $\mathsf{Send}(\Pi_B^j, N_a||A)$: When this query is asked, $\mathbf{G}_1$ makes an input $m = [\mathsf{S}, N_a||A||B, \emptyset]$ and executes $\mathsf{Device}(\Pi_B^j, m, \cdot)$ query and obtains an answer $[N_a||A||B||N_b, t_b, k_1, k_2]$ for a random number N_b. Then $\mathbf{G}_1$ finally outputs $(N_a||A||B||N_b, t_b)$ as an answer.
  - $\mathsf{Send}(\Pi_B^j, m_A||t_a)$: When this query is asked, $\mathbf{G}_1$ makes $m = [\mathsf{S}, m_A, t_a]$. $\mathbf{G}_1$ does simulate inside steps for B's device by asking $\mathsf{Device}(\Pi_B^j, m, \cdot)$ query. If t_a is equal to $F_d(m_A||B_{pub})$ then $\mathbf{G}_1$ obtains an answer $[m_A||N_b', t_b', k_1, k_2]$ where $t_b' = F_d(m_A||N_b'||A_{pub}), k_1 = F_d(m_A||N_b'||1), k_2 = F_d(m_A||1)$ for a random number N_b'. $\mathbf{G}_1$ establishes a session key $sk = k_2||k_B$ where k_B comes from previous simulation of $\mathsf{Send}(\Pi_B^j, N_a||A)$ query. If $\mathsf{Device}(\Pi_B^j, m, \cdot)$ outputs FAIL then it directly outputs an authentication failure.
  - $\mathsf{Send}(\Pi_A^i, m_B||t_b)$: When this query is asked, $\mathbf{G}_1$ initializes an input $m = [\mathsf{S}, m_B, t_b]$ and asks $\mathsf{Device}(\Pi_A^i, m, \cdot)$ in order to simulate the steps for A's device. If t_b is equal to $F_d(m_B||A_{pub})$ then $\mathbf{G}_1$ finally gets m_A, t_a, k_1, k_2 where $t_a = F_d(m_B||N_a'||B_{pub}), k_1 = F_d(m_B||N_a'||1), k_2 = F_d(m_B||1)$. Otherwise if the output is FAIL then terminates the protocol as an authentication failure. $\mathbf{G}_1$ establishes $sk = k_1||k_2$ and outputs m_A, t_a as an final answer on this Send query.
- Other queries: They are simulated as follows.
  - $\mathsf{Execute}(\Pi_A^i, \Pi_B^j)$ query: Through the inputs and outputs of Send queries, the honest transcripts $(N_a, A, m_B, t_b, m_A, t_a)$ between Π_A^i and Π_B^j are returned.
  - $\mathsf{Reveal}(U_i^t)$ query: $\mathbf{G}_1$ returns the established session key sk.
  - $\mathsf{Corrupt}(\Pi_A^i)$, $\mathsf{Corrupt}(\Pi_B^j)$ query: A's (or B's) long-term keys in each platform are returned to $\mathcal{A}$.
  - $\mathsf{CorruptD}(\Pi_A^i)$, $\mathsf{CorruptD}(\Pi_B^j)$ query: A's (or B's) long-term keys in each device are returned to $\mathcal{A}$.
  - Test query. For a Test query, $\mathbf{G}_1$ flips a coin to decide d. If $d = 1$, $\mathbf{G}_1$ returns a real session key sk. Otherwise, $\mathbf{G}_1$ returns a random key.

A collision event Col is analyzed in this game. It is an event that a same session key is generated for two arbitrary sessions due to the duplication of messages from simulation of queries. That is, if the random values N_a', N_a, N_b' are duplicated for arbitrary sessions, then protocol's messages $m_A(= N_a||A||B||N_b||\tilde{N}_a)$, $m_B(= N_a||A||B||N_b)$ are duplicated. Then it makes a same session key $sk = F_d(m_B||1)||F_d(m_A||1)$

for two sessions. This happens within negligible probability $\frac{(q_2+q_s)^6}{2^{3l_r+3}}$ by birthday paradox. We also exclude duplicative cases from simulating a random oracle H_1, which is within $\frac{q_{h_1}^2}{2^{l_{h_1}+1}}$. Thus we have

$$\left|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]\right| \leq \Pr[\mathsf{Col}] = \frac{(q_e+q_s)^6}{2^{3\cdot l_r+3}} + \frac{q_{h_1}^2}{2^{l_{h_1}+1}} \quad (2)$$

where l_r is the size of each random number.

Game $\mathbf{G}_2$*:* In this game, we replace an output of ideal PUF with a random value. That is, on input x, a random y is chosen from $\{0,1\}^{l_o}$. Finally y is returned as an answer for $\mathsf{PUF}_a(x)$ (or $\mathsf{PUF}_b(x)$). For valid simulation, we also maintain list tables $\mathcal{L}_{P_a}$, $\mathcal{L}_{P_b}$ like a random oracle H_1. On a query x_a to $\mathsf{PUF}_a(\cdot)$, if a record (x_a, y_a) appears in $\mathcal{L}_{P_a}$, then y_a is outputted as an answer. Otherwise, a new $y_a \in \{0,1\}^{l_o}$ is randomly chosen and returned. The record (x_a, y_a) is added to the list $\mathcal{L}_{P_a}$. The same applies to $\mathcal{L}_{P_b}$.

It is straightforward to construct an adversary $\mathcal{A}_{prf}$ by using $\mathcal{A}$ that is able to distinguish two games. In $\mathbf{G}_1$ an ideal PUF is used, but in $\mathbf{G}_2$ a randomly chosen value is used for simulating PUF. $\mathcal{A}_{prf}$ just returns $\mathcal{A}$'s output bit d' as an guessing bit b'. Then we have

$$\begin{aligned}\epsilon_{puf} &= |Pr[\mathsf{Exp}^{puf_1}_{\mathcal{A}_{puf}}(k)=1] - Pr[\mathsf{Exp}^{puf_0}_{\mathcal{A}_{puf}}(k)=1]| \\ &= |\Pr[b=b'|\mathsf{PUF}_a(x), \mathsf{PUF}_b(x)] \\ &\quad - \Pr[b=b'|y_a \leftarrow \{0,1\}^{l_o}, y_b \leftarrow \{0,1\}^{l_o}]| \\ &\geq |\Pr[d=d'|G_1] - \Pr[b=b'|G_2]| \\ &= \left|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_2]\right| \end{aligned} \quad (3)$$

Game $\mathbf{G}_3$*:* In this game we consider an event VOD that $\mathcal{A}$ produces a valid output on A's (or B's) device for making a fresh session key. According to the definition of freshness, we classify two subcases, as follows.

- $\mathsf{Device}(\Pi_A^i, m, \cdot)$, $\mathsf{Device}(\Pi_B^j, \cdot, x)$: First, we suppose that $\mathcal{A}$ can access $\mathsf{Device}(\Pi_A^i, m, \cdot)$ and $\mathsf{Device}(\Pi_B^j, \cdot, x)$ queries. However, without S, $\mathcal{A}$ cannot compute d through $\mathsf{Device}(\Pi_B^j, \cdot, x)$ query. $\mathcal{A}$ may freely guess $\widetilde{\mathsf{S}}$ and capture m_B, t_b from protocol messages. $\mathcal{A}$ finally makes $m' = \widetilde{\mathsf{S}}||m_B||t_b$ and asks $\mathsf{Device}(\Pi_A^i, m', \cdot)$, $\mathsf{Device}(\Pi_B^j, m', \cdot)$ to get k_1, k_2. Since $\widetilde{\mathsf{S}}$ is not S, $\mathsf{Device}(\Pi_A^i, m', \cdot)$, $\mathsf{Device}(\Pi_B^j, m', \cdot)$ answer an output FAIL. That is, $\mathcal{A}$ should correctly guess S first, which is negligible probability within $\frac{q_d^2}{2^{l_s}}$.
- $\mathsf{Corrupt}(\Pi_A^i)$, $\mathsf{Corrupt}(\Pi_B^j)$: Second, let's suppose that $\mathcal{A}$ can access $\mathsf{Corrupt}(\Pi_A^i)$ and $\mathsf{Corrupt}(\Pi_B^j)$ queries. Without querying $\mathsf{Device}(\Pi_A^i, \cdot, \cdot)$, $\mathsf{Device}(\Pi_B^j, \cdot, \cdot)$ queries, the secret S only is no use for making a valid outputs of devices. Thus, in order to impersonate devices, $\mathcal{A}$ may try to compute d. That is, $\mathcal{A}$ asks CorruptD query first and obtains $A_{pub} = g^{r_a}, B_{pub} = g^{r_b}$ to gain $d = H_2(g^{r_a r_b})$. However, these values never help $\mathcal{A}$ compute d. If this case happens it is easy to construct a polynomial time $\mathcal{A}_{cdh}$ that forges $g^{r_a r_b}$. Hence it is negligible probability ϵ_{cdh} by CDH assumption. Thus, the last way for establishing a valid sk would be just to guess k_1, k_2, which are outputs of pseudo random function, which is negligible, $\frac{q_d^4}{2^{2L}}$.

Two games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable unless VOD does not happen.

$$\left|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]\right| \leq \Pr[\mathsf{VOD}] \leq \frac{q_d^2}{2^{l_s}} + \frac{q_d^4}{2^{2L}} + \epsilon_{cdh} \quad (4)$$

Game $\mathbf{G}_4$*:* In this game, we consider an output of pseudo random function F_d. We replace F_d with a uniformly random function F from the set of all functions U_{F_K}. Then the session key $sk = F_d(m_A||0)||F_d(m_B||1)$ is simulated as $sk = F(m_A||0)||F(m_B||1)$. The simulated sk is uniformly random and is independent from all possible session keys. Thus we have

$$\Pr[\mathsf{S}_4] = \frac{1}{2} \quad (5)$$

It is clear to show that the difference between two games is within ϵ_{prf} as follows.

$$\begin{aligned}\epsilon_{prf} &= |\Pr[b=b'|sk = F_d(m_A||1)||F_d(m_B||1)] \\ &\quad - \Pr[b=b'|sk = F(m_A||1)||F(m_B||1)]| \\ &= \left|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_4]\right| \end{aligned} \quad (6)$$

From the equations (1)∼(6), we finally obtain the following main result.

$$\begin{aligned}\Pr[\mathsf{S}_0] \leq\ & \frac{(q_e+q_s)^6}{2^{3\cdot l_r+3}} + \frac{q_{h_1}^2}{2^{l_{h_1}+1}} \\ & + \epsilon_{puf} + \frac{q_d^2}{2^{l_s}} + \frac{q_d^4}{2^{2L}} + \epsilon_{cdh} + \frac{1}{2} \end{aligned} \quad (7)$$

□

VII. PERFORMANCE ANALYSIS AND COMPARISON

Let's analyze the performance of our scheme. First of all, we note that our scheme is based on a different PUF setting while others are based on single PUF setting. Nevertheless, Table 1 shows that our scheme preserves efficiency with security as compared with other schemes. We classify the most relevant PUF-based schemes into two categories; a general RFID setting and a normal two party setting (IoT, multi-factor, AKE).

First, the RFID setting assumes an unbalanced network where a reader with back end server desires to identify a tag with limited resources, and it usually focuses on mutual authentication rather than an authenticated key exchange. As shown in Table 1, the relevant PUF-based RFID protocols do not support a key agreement (exchange) either while our scheme guarantees authenticated key agreement. The forward secrecy here is the security for the past agreed session key after long-term secrets are revealed. In other words, the forward secrecy cannot be discussed without supporting a session key agreement. Hence Table 1 shows that all PUF-based RFID schemes doesn't provide forward secrecy. On the other hand, our scheme guarantees the forward secrecy

and mutual authentication with authenticated key exchange. Furthermore, our scheme needs the least three communication flows while others [27], [44] need four or five flows. In view of the communication size, the LNB [29], KYWG [27] schemes are efficient than our scheme, but they do not support an authenticated key agreement or mutual authentication.

Second, in normal two-party setting, FBA [17] and AS [3] schemes just needs 2 flows, but they aim for uniliteral authentication. Only the RMKWD [36] scheme needs less communication size (3ł_r) than ours, but it only guarantees uniliteral authentication. Therefore our scheme is superior than others in aspect of both efficiency and security.

One security concern is the privacy property. It stems from our security model that any adversary can access user's Device and PUF on behalf of the participants. That is, to ask Device query, an adversary should take an identifier as inputs in our security model. That is, accessing Device oracles by an adversary means that user's identifier is already known to the adversary. In our security model, therefore, it would be a contradiction to be able to satisfy identifier protection. On the other hand, in aspect of efficiency, our scheme needs three communication flows for mutual authentication. Also, our scheme less requires the total size of communication than others. In conclusion, our scheme is the first PUF-to-PUF AKE scheme that guarantees efficient communication cost, session key security, and forward secrecy with mutual authentication.

VIII. EXTENSIONS AND FUTURE WORKS

There may be possible extensions and variants of the original scheme. Here we discuss two obvious settings such as password and asymmetric, as follows.

- *Password setting:* A common password (or different passwords) between A and B can be considered as an alternative authentication factor. For instance, in our scheme, a password pwd can be used as substitute for a secret S, which results in convenience and efficiency due to its memorable property. However, one serious threat is that the password can be guessed in an off-line way under the Device oracles. For example, let's suppose that the secret is replaced with pwd. Then $\mathcal{A}$ may obtain a session key $sk = F_d(m_A||1)||F_d(m_B||1)$ for valid transcripts (m_B, t_b, m_A, t_a). $\mathcal{A}$ guesses a password pwd' and asks $\mathsf{Device}(\Pi_B^j, pwd'||m_B||\emptyset)$. It receives answers $[m_b||N_b, t_b'(= F_{d'}(m_b||N_b||A_{pub}))$, $k_1'(= F_{d'}(m_b||N_b||1))$, $k_2(= f_{d'}(m_B||1))]$ for $d' = H_1((A_{pub})^{r_b} \bmod p)$. Among answers, $\mathcal{A}$ just checks if $k_2(= F_d'(m_b||N_b||1))$ is equal to k_2 then reduces the candidates from a dictionary of password, as many as $\mathcal{A}$ wants, in an off-line way. It doesn't seem to be an easy task to securely design for PUF to have an input of password under assumption of Device oracle. It remains an future work.
- *Asymmetric setting:* An asymmetric approach like the Diffie-Hellman key agreement can be considered for guaranteeing forward secrecy. As discussed earlier, if we suppose a strong adversary $\mathcal{A}$ that accesses a secret S and two Device oracles, then $\mathcal{A}$ is able to compute all past session keys. This is a natural result, but if we extend our scheme into asymmetric setting where a pair of public/private keys are assigned to participants, then forward secrecy may be guaranteed against $\mathcal{A}$. For this end, one easy solution is to design that A and B exchange $g^x \bmod p$, $g^y \bmod p$ in an authentication phase and make a final session key $sk = H(g^x||g^y||F_d(m_A||1)||F_d(m_B||1)||g^{ay+bx})$ where (a, g^a), (b, g^b) are private/public key pairs for A and B. Then $\mathcal{A}$ should face the intractability of Diffie-Hellman problem (g^{ay} or g^{bx}) in order to compute sk even if $\mathcal{A}$ accesses $\mathsf{Device}(\Pi_A^i, m)$ and $\mathsf{Device}(\Pi_B^j, m)$ oracles with a secret S. This is a simple suggestion on authentication phase, but it still remains a future work to design a full version of protocol secure against new security model with provable security. Further, more strengthened security model [13], [22], [28] such as CK, CK+, eCK, can be considered in asymmetric setting. In these case, considerable changes or modifications may be required for a secure and efficient construction.

IX. CONCLUDING REMARKS

In this paper we presented a new authentication setting where two end users holding two authentication factors (own PUF-embedded device and a long-term secret) desire to authenticate each other with an agreed session key over a distributed network. In establishing a new model, we firstly explored a new concept of Device oracle which models that an adversary picks up an arbitrary device and attempts to input a message and obtains its output from the device. Under a new setting, we newly presented a distinct PUF-embedded AKE with provable security.

ACKNOWLEDGMENT

We are greatly thankful to anonymous reviewers for helpful suggestions and comments.

REFERENCES

[1] M. Akgün and M. U. Çağlayan, "Providing destructive privacy and scalability in RFID systems using PUFs," *Ad Hoc Netw.*, vol. 32, pp. 32–42, Sep. 2015.
[2] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls, "Memory leakage-resilient encryption based on physically unclonable functions," in *Proc. ASIACRYPT*, in Lecture Notes in Computer Science, vol. 5912, 2000, pp. 685–702.
[3] M. N. Aman and B. Sikdar, "ATT-Auth: A hybrid protocol for industrial IoT attestation with authentication," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5119–5131, Dec. 2018. doi: 10.1109/JIOT.2018.2866623.
[4] A. Braeken, "PUF based authentication protocol for IoT," *Symmetry*, vol. 10, no. 8, p. 352, 2018. doi: 10.3390/sym10080352.
[5] J. W. Byun, "An efficient multi-factor authenticated key exchange with physically unclonable function," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, 2019, pp. 1–4. doi: 10.23919/ELINFOCOM.2019.8706400.
[6] J. W. Byun, "A generic multifactor authenticated key exchange with physical unclonable function," *Secur. Commun. Netw.*, vol. 2019, Mar. 2019, Art. no. 5935292. doi: 10.1155/2019/5935292.
[7] R. Bassil, W. El-Beaino, A. Kayssi, and A. Chehab, "A PUF-based ultra-lightweight mutual-authentication RFID protocol," in *Proc. 6th Int. Conf. Internet Technol. Secured Trans.*, 2011, pp. 495–499.

[8] C. Brzuska, M. Fischlin, H. Schröder, and A. Katzenbeisser, "Physically uncloneable functions in the universal composition framework," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 6841. Berlin, Germany: Springer, 2011, pp. 51–70.

[9] H. Busch, S. Katzenbeisser, and P. Baecher, "PUF-based authentication protocols—Revisited," in *Proc. WISA*, in Lecture Notes in Computer Science, vol. 5932. Berlin, Germany: Springer-Verlag, 2009, pp. 296–308.

[10] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. Eurocrypt*, in Lecture Notes in Computer Science, vol. 1807. Berlin, Germany: Springer-Verlag, 2000, pp. 139–155.

[11] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, pp. 67-1–67-25, 2017.

[12] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, and R. S. Chakraborty, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 3, pp. 424–437, May/Jun. 2019.

[13] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Eurocrypt*, in Lecture Notes in Computer Science, vol. 2045. Berlin, Germany: Springer-Verlag, 2001, pp. 453–474.

[14] W. Che, F. Saqib, and J. Plusquellic, "PUF-based authentication," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 337–344.

[15] M. van Dijk and U. Rührmair, "Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results," Cryptol. ePrint Arch., Tech. Rep. 2012/228, 2012. [Online]. Available: https://eprint.iacr.org/2012/228

[16] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 3027. Berlin, Germany: Springer, 2004, pp. 523–540.

[17] K. B. Frikken, M. Blanton, and M. J. Atallah, "Robust authentication using physically unclonable functions," in *Proc. ISC*, in Lecture Notes in Computer Science, vol. 5735. Berlin, Germany: Springer, 2009, pp. 262–277.

[18] P. Gope, J. Lee, and T. Q. S. Quek, "Lightweight and practical anonymous authentication protocol for RFID systems using physically unclonable functions," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2831–2843, Nov. 2018.

[19] P. Gope and B. Sikdar, "Privacy-aware authenticated key agreement scheme for secure smart grid communication," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3953–3962, Jul. 2019. doi: 10.1109/TSG.2018.2844403.

[20] P. Gope and B. Sikdar, "Lightweight and privacy-preserving two-factor authentication scheme for IOT devices," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 580–589, Feb. 2019. doi: 10.1109/JIOT.2018.2846299.

[21] D. Hofheinz and E. Kiltz, "Programmable hash functions and their applications," *J. Cryptol.*, vol. 25, no. 3, pp. 484–527, 2012.

[22] H. Krawczyk, "HMQV: A high-performance secure Diffie–Hellman protocol," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 3621. Berlin, Germany: Springer, 2005, pp. 546 566.

[23] S. D. Kaul and A. K. Awasthi, "Privacy model for threshold RFID system based on PUF," *Wireless Pers. Commun.*, vol. 95, no. 3, pp. 2803–2828, 2017.

[24] S. Kardas, M. Akgün, M. S. Kiraz, and H. Demirci, "Cryptanalysis of lightweight mutual authentication and ownership transfer for RFID systems," in *Proc. Workshop Lightweight Secur. Privacy, Devices, Protocols, Appl.*, 2011, pp. 20–25.

[25] S. Kardaş, S. Çelik, M. Yıldız, and A. Levi, "PUF-enhanced offline RFID security and privacy," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 2059–2067, 2012.

[26] S. Katzenbeisser, U. Kocabaş, V. van der Leest, A.-R. Sadeghi, G.-J. Schrijen, C. Wachsmann, and H. Schröder, "Recyclable PUFs: Logically reconfigurable PUFs," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 6917. Berlin, Germany: Springer, 2011, pp. 374–389.

[27] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight mutual authentication and ownership transfer for RFID systems," in *Proc. IEEE Conf. INFOCOM*, Mar. 2010, pp. 251–255.

[28] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *Provable Security* (Lecture Notes in Computer Science) vol. 4784. Berlin, Germany: Springer, 2007, pp. 1–16.

[29] K. Lee, J. G. Nieto, and C. Boyd, "A state-aware RFID privacy model with reader corruption," in *Cyberspace Safety and Security* (Lecture Notes in Computer Science), vol. 7672. Berlin, Germany: Springer, 2012, pp. 324–338.

[30] P. K. Maurya and S. Bagchi, "A secure PUF-based unilateral authentication scheme for RFID system," *Wireless Pers. Commun.*, vol. 103, no. 2, pp. 1699–1712, 2018. doi: 10.1007/s11277-018-5875-2.

[31] S. Mauw and S. Piramuthu, "A PUF-based authentication protocol to address ticket-switching of RFID-tagged items," in *Security and Trust Management* (Lecture Notes in Computer Science), vol. 7783. Berlin, Germany: Springer, 2013, pp. 209–224.

[32] D. Moriyama, S. Matsuo, and M. Yung, "PUF-based RFID authentication secure and private under memory leakage," Cryptol. ePrint Arch., Tech. Rep. 2013/712, 2013. [Online]. Available: https://eprint.iacr.org/2013/712

[33] U. Rührmair, "Oblivious transfer based on physical unclonable functions," in *Trust and Trustworthy Computing* (Lecture Notes in Computer Science), vol. 6101. Berlin, Germany: Springer, 2010, pp. 430–440.

[34] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Proc. CHES*, in Lecture Notes in Computer Science, vol. 7428, 2012, pp. 302–3019.

[35] A. C. D. Resende, K. Mochetti, and D. F. Aranha, "PUF-based mutual multifactor entity and transaction authentication for secure banking," in *Proc. LightSec*, in Lecture Notes in Computer Science, vol. 9542, 2015, pp. 77–96.

[36] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 37–49, Mar. 2014.

[37] A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "PUF-enhanced RFID security and privacy," in *Proc. SECSI*, Berlin, Germany, 2010.

[38] U. Rührmair, C. Jaeger, and M. Algasinger, "An attack on PUF-based session key exchange and a hardware-based countermeasure: Erasable PUFs," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science) vol. 7035. Berlin, Germany: Springer, 2011, pp. 190–204.

[39] A. C. D. Resende, K. Mochetti, and D. F. Aranha, "PUF-based mutual multifactor entity and transaction authentication for secure banking," in *Proc. ISC*, in Lecture Notes in Computer Science, vol. 5735. Berlin, Germany: Springer, 2009, pp. 262–277.

[40] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," Cryptol. ePrint Arch., Rep. 2004/332, 2004. [Online]. Available: https://eprint.iacr.org/2004/332

[41] A. Sadr and M. Zolfaghari-Nejad, "Physical unclonable function (PUF) based random number generator," *Adv. Comput., Int. J.*, vol. 3, no. 2, pp. 139–145, 2012.

[42] P. Tuyls and B. Škorić, "Strong authentication with physical unclonable functions," in *Security, Privacy, and Trust in Modern Data Management*. Berlin, Germany: Springer, 2007, pp. 133–148.

[43] S. Vaudenay, "On Privacy Models for RFID," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science) vol. 4833. Berlin, Germany: Springer, 2007, pp. 68–87.

[44] H. Xu, J. Ding, P. Li, F. Zhu, and R. Wang, "A lightweight RFID mutual authentication protocol based on physical unclonable function," *Sensors*, vol. 18, no. 3, p. 760, 2018.

[45] J. Zhang, X. Tan, X. Wang, A. Yan, and Z. Qin, "T2FA: Transparent two-factor authentication," *IEEE Access*, vol. 6, pp. 32677–32686, 2018. doi: 10.1109/ACCESS.2018.2844548.

JIN WOOK BYUN received the B.S. degree from the Department of Computer Science, and the M.S. and Ph.D. degrees from the Graduate School of Information Security, Korea University, in 2001, 2003, and 2006, respectively. He has joined the Faculty of the Department of Information and Communication, Pyeongtaek University, in 2008. His research interests include the design of cryptographic protocol, authenticated key exchange, and database security.