Forward-Secure Authenticated Symmetric Key Exchange Protocol: New Security Model and Secure Construction

Suvradip Chakraborty[1], Goutam Paul[2(✉)], and C. Pandu Rangan[1]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai, India
{suvradip1111,prangan55}@gmail.com
[2] Cryptology and Security Research Unit (CSRU),
R. C. Bose Centre for Cryptology and Security,
Indian Statistical Institute, Kolkata, India
goutam.paul@isical.ac.in

Abstract. While a lot of work has been done on the design and security analysis of PKI-based authenticated key exchange (AKE) protocols, very few exist in the symmetric key setting. The first provably secure symmetric AKE was proposed by Bellare and Rogaway (BR) in CRYPTO 1994 and so far this stands out as the most prominent one for symmetric key setting. In line with the significant progress done for PKI based system, we propose a stronger model than the BR model for symmetric key based system. We assume that the adversary can launch active attacks. In addition, the adversary can also obtain long term secret keys of the parties and the internal states of parties by getting access to their ephemeral secrets (or internal randomness) by means of appropriate oracle queries. The salient feature of our model is the way we handle active adversaries even in the test session.

We also design a symmetric key AKE construction that is provably secure against active adversaries in our new model using weak primitives. Dodis et al. (EUROCRYPT 2012) used weak Pseudo Random Functions (wPRF) and weak Almost-XOR Universal hash function family (wAXU) to design a three-pass one-sided authentication protocol in the symmetric key paradigm. A direct application of their techniques yields a four-pass (two-round) symmetric key AKE protocol with mutual authentication. Our construction uses particular instances of these weak primitives and introduces a novel technique called input-swapping to achieve a three-pass symmetric key AKE protocol with mutual authentication resisting active attacks (even in the test session). Our construction is proven secure in the Random oracle Model under the DDH assumption.

Keywords: Authenticated key exchange · Input swapping · Random oracle · Key evolving · Perfect forward secrecy · Weak pseudo random functions · Weak almost universal hash functions

M.-H. Au and A. Miyaji (Eds.): ProvSec 2015, LNCS 9451, pp. 149–166, 2015.
DOI: 10.1007/978-3-319-26059-4_8

1 Introduction

Key exchange protocols allow two parties to establish secure communication over an untrusted network by setting up shared keys. Authenticated Key Exchange protocol (AKE) allows two parties not only to share a session secret key but also to mutually authenticate each other. After the key is securely established between the two parties, the key is used for encrypting messages among the two parties.

Most symmetric key encryption systems assume that the two parties share a common secret. Thus, before exchanging the actual data, the two parties first need to establish such a shared secret. For this, they usually first run a *key agreement* protocol over the public channel to agree on a symmetric key. We refer the established key by key agreement protocol as Long Term key (LTK) between a pair of parties.

In symmetric key set up, key exchange protocols broadly fall under two categories: *i. Server-based* key exchange protocols where the presence of a trusted server is assumed and all the parties have a LTK shared with the server; all the communication between the two parties take place via the server [NS78, OR87, GNY90] and *ii. Server-less* key exchange protocols where the presence of a trusted server is not required but it is assumed that the communicating parties share a LTK between themselves [Sat90, BR94, CJ97]. In this paper we focus on Server-less two party symmetric key exchange protocol.

While the communicating parties may use their LTKs to securely exchange messages for confidentiality or authenticity, it is generally considered a bad practice. This is because even if the pre-shared keys or LTKs have good entropy, the key tends to weaken with use; this is the reason why session keys are used instead and changed frequently. Hence, there is a need for key derivation based on pre-shared keys.Generally, for each session, the parties generate a session randomness, called *ephemeral secret key* and use that to generate the session key. This session key can be applied to subsequent network systems including live conference, online video games, collaborative work spaces and much more.

The Diffie-Hellman protocol [DH76] was the first key-exchange protocol not requiring a pre-shared static secret between the parties. It does not however enforce authentication between the parties, and is therefore vulnerable to man-in-the-middle attacks. After that, many *authenticated key-exchange* (AKE) protocols were proposed [JKL04, BLL07, CL08, CY08, LCZ07]; either they are insecure or their security is proved in an ad-hoc manner and later on some attacks would be presented.The attacked systems were either modified in a minor way to just overcome the proposed attack or abandoned altogether. This prompted the need for formal models for AKE problem and robust protocols for AKE whose security is proved formally in these models. Starting from the seminal work of Bellare and Rogaway [BR94], progressive stronger models were proposed by several authors such as [LMQ+03, Ust08, Kra05, SEVB10, MU08, CK01, LLM07, Sho99]. However all these are defined for the PKI based systems.

Surprisingly, after the first formal model for AKE was proposed by Bellare and Rogaway [BR94] for symmetric key system, no further progress has

happened in refining the system for symmetric key settings considering stronger classes of adversaries. Several authors have presented protocols for AKE in the symmetric key set up such as [NS78,OR87] etc. However, none of them were proved and they were shown to be susceptible to various classes of attacks [Boy90,CJ97,BGH+92]. In fact many of the ISO-9798 family of protocols were shown to vulnerable to various classes of attacks [BCM13].

Our first contribution is that we propose a new model for a symmetric key AKE that allows the adversary to have far more power than the BR model [BR94]. More specifically, the adversary can obtain session randomness of party, long term keys of a parties. In particular we allow the adversary to obtain all information other than those which allow him to trivially compute the session key for a particular session (as defined by the freshness condition in Sect. 3.3). Moreover we allow the adversary to be fully active even in the test session. We define what it means for a symmetric key AKE to be secure in our new model.

We then give a construction of a three-pass symmetric key exchange protocol from *weak Pseudo Random Functions* (wPRF) and *weak Almost-XOR Universal hash function family* (wAXU). Dodis et al. [DKPW12] showed how to construct a three-pass authentication protocol from wPRF secure against *active* attacks. If we use the techniques of [DKPW12] to obtain mutual authentication, it will result in a four-pass (two-round) symmetric key AKE. We improve upon this direct application of their scheme by showing how to obtain a three-pass actively secure symmetric key AKE using secure instances of the same primitives which provides mutual authentication. For this, we introduce a new technique, called *input swapping*. Ours is the *first* construction that achieves security against fully active adversaries and at the same time provide mutual authentication of both parties in only *three-pass* using weaker primitives than MAC or PRF. Our proposed construction is also efficient and is secure under the *Decisional Diffie Hellman* (DDH) assumption in the Random Oracle (RO) model.

Our construction is also *forward secure*. So even if an adversary can get the long term secret key (LTK) between two parties, he cannot infer the previously established session keys between them. The concept of *perfect forward secrecy* was first defined by Gunther [Gün90] and used in protocols like Station-to-Station (STS) [DVOW92], SASI [Chi07]. These protocols update the LTKs at regular interval of time and irreversibly, i.e., if an adversary gets hold of a LTK K_t at time interval t say, he cannot infer the previous LTKs before time period t, since it will require the adversary to break the one-wayness of some function. These schemes are called *Key-Evolving Schemes* (KES). The time interval after which the LTKs should be updated depends from application to application depending on the level of security they want to achieve. They can be updated after each session, or they may be updated after every ϱ interval of time. The first approach is generally not followed due to the concurrent nature of AKE protocols. It may happen that multiple instances of the same protocol is running between different parties. If the LTKs are updated after each session, it may cause some other sessions to abort. So generally the second approach is followed which is far more suitable for distributed setting like the Internet. However on

the downside, if we use the second approach for key evolution we do not achieve perfect forward secrecy in its entirely, whereas if we update the LTKs after each session we achieve perfect forward secrecy. The faster the LTKs are evolved more forward secrecy is achieved. We also evolve the LTKs at regular time intervals or *epoch*. The duration of each epoch after which the LTK needs to be updated or evolved depends on the application. If the level of security needs to be high, then the key refreshment needs to be done frequently. However it should not be the case that the key refreshment rate gets greater than the rate at which the session key is actually established between the parties. In some sense our constructions are also *key-evolving*. So we can achieve perfect forward secrecy due to the key evolving nature of our protocol and also due to the fact that in each session the parties choose independent randomness.

1.1 Our Contributions

To summarize what we have said earlier in the previous section our main contributions are as follows:

1. We propose a new model for a symmetric forward-secure AKE that is more powerful than the BR model [BR94]. In addition, our model has the capability to handle active adversaries even in the test session.
2. We propose a concrete three-pass symmetric key AKE protocol secure in our new model withstanding active adversaries. The security of this protocol is proved under the *Decisional Diffie Hellman* (DDH) assumption in the random oracle model.
3. Our construction uses much weaker primitives like *weak Pseudo Random Functions* (wPRF) and *weak Almost-XOR Universal hash function family* (wAXU) as in [DKPW12]. However, with the introduction of a new technique which we call *input swapping*, combined with these primitives, we can complete our protocol in three-passes.
4. Our protocol also achieves *perfect forward security*. We achieve this by a key evolving strategy and also sampling independent randomness for each party in each session.

2 Preliminaries

In this section, we define all the notations we would be using throughout the paper. We also provide some standard definitions and state the complexity assumptions required for our constructions.

2.1 Notations

The set of integers modulo an integer p is denoted by $\mathbb{Z}_p$. Let $[n]$ denote the set of integers $\{1, 2, \ldots, n\}$ where $n \in \mathbb{N}$, the set of all natural numbers. For a set X, $x \in_R X$ denotes that x is randomly sampled from the set X. We denote by

λ the security parameter and it will be given in unary to the algorithms. All the algorithms will run in time polynomial in the size of the security parameter. Let $\{0,1\}^*$ denote the set of all binary strings and $\{0,1\}^n$ denote the set of all binary strings of length n. The length of a string $x \in \{0,1\}^*$ is denoted by $|x|$. Let p be a large prime of order λ where $p = 2q + 1$ and q is a prime number. Let $\mathbb{G}$ be a multiplicative subgroup of $\mathbb{Z}_p^*$ with prime order q and let g be a generator for $\mathbb{G}$. A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible* if it vanishes faster than the inverse of any polynomial, i.e., for a constant $c > 0$ and sufficiently large n, $f(n) < n^{-c}$. We define $\mathsf{DLOG}(\mathsf{x})$ to denote the discrete logarithm of x with base g.

2.2 Some Standard Definitions

Definition 1. ***Weak Pseudo-Random Functions(wPRF).*** *A function family $\mathcal{F} = \{f_K\}_{K \in \mathcal{K}} : \mathbb{D} \to \mathbb{F}$ where $\mathcal{K}$ is the key space, is said to be a weak PRF family if for any polynomial-sized k, randomly chosen $f \in_R \mathcal{F}$, and $r_1, r_2, \ldots r_k \in_R \mathbb{D}$, the distribution of $(r_1, f(r_1)), (r_2, f(r_2)), \ldots, (r_k, f(r_k))$ is computationally indistinguishable from the uniform distribution over $(\mathbb{D}, \mathbb{F})^k$, i.e., an adversary for a weak-PRF aims to distinguish a random member of the family from a truly random function after observing a polynomially-bounded number of samples. A wPRF is called (t, Q, ϵ)-wPRF if for a t-time adversary $\mathcal{A}$ making at most Q queries to the function, the advantage in distinguishing the above two distributions is at most ϵ.*

Definition 2. ***Almost Universal Hash Family.*** *A family of keyed hash functions $\{\mathcal{H}_k : \mathbb{D} \to \mathbb{F}\}_{k \in \mathcal{K}}$ is ρ-almost universal if $\forall x_1 \neq x_2 \in \mathbb{D}$, $Pr_{k \in_R \mathcal{K}}[h_k(x_1) = h_k(x_2)] \leq \rho$ holds.*

Definition 3. ***Pairwise Independent Hash Family.*** *A hash family $H : \mathbb{D} \to \mathbb{F}$ is called pairwise independent if $\forall x_1, x_2 \in \mathbb{D}$ and $x_1 \neq x_2$, and $y_1, y_2 \in \mathbb{F}$, $Pr_{h \in_R H}[h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{|\mathbb{F}|^2}$.*

Definition 4. ***Weak-Almost XOR-Universal (wAXU) hash family.*** *A family of keyed hash functions $\{\mathcal{H}_k : \mathbb{D} \to \mathbb{F}\}_{k \in \mathcal{K}}$ is δ-wAXU if for $x_1, x_2 \in_R \mathbb{D}$ with $x_1 \neq x_2$, and a $y \in \mathbb{F}$, we have $Pr_{k \in_R \mathcal{K}}[h_k(x_1) \oplus h_k(x_2) = y] \leq \delta$.*

If $\forall x_1 \neq x_2$ and for any $y \in \mathbb{F}$, $Pr_{k \in_R \mathcal{K}}[h_k(x_1) \oplus h_k(x_2) = y] \leq \delta$ holds, then it is called δ-AXU. If $\delta = \frac{1}{|\mathbb{F}|}$, it is called (perfectly) XOR-universal.

Without loss of generality we can assume that the output of all the functions can be embedded into a finite field. Example if the output size of a wPRF is n bits, one can embed the output of the wPRF into a finite field of size 2^n. The simplest way this can be done is considering the function output $t \in \{0,1\}^n$ as a polynomial in a finite field $\mathbb{F} = (\mathbb{Z}_2^n, +, \times)$ for suitable defined addition and multiplication operations.

2.3 Complexity Assumption

In this section, we present a brief overview of the hard problem assumption.

Definition 5. ***Decisional Diffie-Hellman Problem (DDH)*** *- Given* $(g, g^{\alpha}, g^{\beta}, h) \in \mathbb{G}^4$ *for unknown* $\alpha, \beta \in \mathbb{Z}_q^*$*, where* $\mathbb{G}$ *is a cyclic prime order multiplicative group with* g *as a generator and* q *the order of the group, the DDH problem in* $\mathbb{G}$ *is to check whether* $h \stackrel{?}{=} g^{\alpha\beta}$*.*

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ *in solving the DDH problem in* $\mathbb{G}$ *is defined as*

$$Adv_{\mathcal{A}}^{DDH} = |Pr\left[\mathcal{A}(g, g^{\alpha}, g^{\beta}, g^{\alpha\beta}) = 1\right] - Pr\left[\mathcal{A}(g, g^{\alpha}, g^{\beta}, h) = 1\right] \mid \alpha, \beta \in \mathbb{Z}_q^*|$$

The DDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$*, the advantage* $Adv_{\mathcal{A}}^{DDH}$ *is negligibly small.*

3 Our Security Model

In order to define what is meant by the security of an authenticated symmetric key exchange protocol, we need to formally define the security model and to define the powers of an adversary in this model. We assume there are n parties $P_1, \ldots, P_n$ each modeled by a probabilistic polynomial time Turing machine (PPTM). We assume the parties $P_1, \ldots, P_n$ are connected over point-to-point links over which the messages can be exchanged between them. Since our AKE is *key evolving*, the LTKs between the parties are fixed for a particular epoch only and they are refreshed at the expiration of each epoch. In particular, the l^{th} epoch is denoted by T_l and it represents the time interval $[a_{l-1}, a_l]$ $\forall l \geq 1$. In each epoch T_l, the parties P_i and P_j will have a LTK shared among them denoted by $LTK_{ij}^{(l)}$ $\forall l \geq 1$. We denote the current epoch as T_t throughout the paper. We assume that at the beginning of each epoch, the LTKs of all the parties are updated simultaneously. Without loss of generality, we assume throughout the paper that when two parties communicate with each other, they do so in the same epoch. We now define the concept of a session and define the powers of an adversary in our model. Next, we define the concept of partner session in this setting and what it means for a session to be fresh. Finally, we define the security of an symmetric AKE protocol in this model.

3.1 Session

A typical AKE protocol between two parties consists of several passes between the parties wishing to establish a common session key. Let us denote the two parties executing an instance of the protocol by A and B. Let us further assume that A is the initiator and B is the responder for this execution of the protocol. The set of all actions carried out by A during the execution of the protocol is called *session* of A. Similarly we define the session corresponding to B.

Every session is uniquely identified by a label called *session identifier*. A session corresponding to the party A consists of a (small) finite number of passes executed by A. In a typical pass executed by A:

1. A may perform some local computations that may depend on the values received from B or locally generated by A and
2. A may communicate certain values to B or *aborts*.

The state of a typical pass, say the i^{th} pass is described by the 5-tuple $\hat{P}_i = (sid_{AB}, A, B, in_i, out_i)$ where A is the *owner* of the session, sid_{AB} is the session identifier, B is the *peer*, in_i denotes the values A had received from B prior to the execution of this pass and out_i is the set of values that A sends to B in this pass.

If A is the initiator of the protocol then the state of the first pass is denoted as $(sid_{AB}, A, B, \_\_\_, out_1)$ because this being the first pass of the session A would not have received any value from B. Since a session comprises of several passes, we define the state of a session corresponding to A as $(sid_{AB}, A, B, IN, OUT, role)$ where A is the owner of the session and B is the peer for the session, $role = \mathcal{I}$ if A is the initiator of the protocol and $role = \mathcal{R}$ if A is the responder of the protocol, $IN = \bigcup_{i=1}^{n} in_i$, where n is the number of passes and in_i is the values received from B in the i^{th} pass and $OUT = \bigcup_{i=1}^{n} out_i$ where out_i denotes the values that A sends to B.

Several sessions may run concurrently at each party. Since a party can have multiple concurrent sessions running within it, it has to maintain the states of all these sessions. However these session states are independent of each other as all the sessions use independent randomness and hence their states are to be stored separately. The shared secret key obtained at the end of this session is called the *session key*. On successful completion of a session, each entity outputs the *session key* and deletes the *session state*. A session may not get completed, and may enter into *abort* state, and in this case no *session key* is generated.

3.2 Adversary

The adversary $\mathcal{A}$ is also modeled as a PPTM which has the capability to launch active attacks in all the session including the test session apart from his eavesdropping capability on the protocol messages. In order to model real life attacks we allow the adversary to obtain more information such as the long term secret key of parties. Since our protocol is key evolving (i.e. the long term secret keys of users changes after a certain lifetime) we allow the adversary to obtain the LTKs at a certain time period. Similarly the adversary can also obtain ephemeral secret key of users during a particular session. In our security model since we allow the adversary to launch active attacks even in the test session, we disallow the adversary to make corrupt queries in the test session for the current epoch. This restriction is quite natural and is an obvious one. In fact, if we allow the adversary to obtain the LTK of the parties in the test session, ephemeral keys of the parties and also allow the adversary to launch active in the test session, it would be impossible to design any protocol as discussed in Sect. 4.3. Thus we need to impose some restriction on the allowed queries when we make the

adversary active in all sessions, including the test session. The information that the adversary gets is modeled by the following oracle queries:

- $Send(P_i, P_j, m, t)$: This query models the capabilities of an adversary to perform the *active man-in-the-middle* attacks. Here t represents the index of the current epoch T_t. The following sequence of actions are assumed to take place:
  1. Case (i) $m \neq 0$
     - Party P_i sends m in this pass to party P_j.
     - If P_j does not abort, it sends a response say r_j to P_i.
     - The value r_j is given to the adversary.
  2. Case (ii) $m = 0$
     - Party P_i initiates a new session with party P_j.
     - If P_j does not abort, it sends a response say r_j to P_i.
     - The value r_j is given to the adversary.
- $LTK\_Reveal(i, j, t)$: This query will return $LTK_{ij}^{(l)}$ $\forall 1 \leq l < t$ where t denotes the index of the current epoch T_t, i.e., it allows the adversary to obtain the LTKs of user P_i and P_j in all the previous epochs excluding the current epoch T_t.
- $Ephemeral\_Reveal(sid_{ij}^{(t)})$: This query reveals the ephemeral secret key of the session $sid_{ij}^{(t)}$, i.e., the value $rand_i$ of the current epoch T_t.
- $SK\_Reveal(sid_{ij}^{(t)})$: The queries reveals the session key of a completed session $sid_{ij}^{(t)}$ for the current epoch T_t.

After some polynomial amount of interactions, A may choose a specific session as *Test* session provided the session is completed and fresh as per Definition 9. We denote the chosen session as $Test(sid_{ij}^{(t)})$. Only one query of this form is allowed.

- $Test(sid_{ij}^{(t)})$: In response to the test query, a bit $b \in \{0, 1\}$ is randomly generated. If $b = 0$, the session key is given to the adversary $\mathcal{A}$. If $b = 1$, a uniformly chosen random value from the set of valid session keys is returned to $\mathcal{A}$.
  After the Test query has been issued, the adversary can continue querying provided that the test session remains fresh. Note that while $\mathcal{A}$ can issue polynomial number of queries like *Send*, LTK_Reveal, $Ephemeral\_Reveal$ queries he can issue only one $Test$ query.

3.3 Few Important Definitions

Before defining the Sym-AKE security notion, we need to define what is meant by *session partnership* or a *matching* session.

Definition 6. ***Session Partnership.*** *Two sessions* $sid_{ij}^{(t)} = \big(P_i, P_j, OUT, IN, role\big)$ *and* $sid_{ji}^{\prime(t)} = (P_j, P_i, OUT', IN', role')$ *are said to be partnered session or matching sessions iff:*

1. $OUT = IN'$ *and* $IN = OUT'$.
2. $role \neq role'$.

We now define the condition for the freshness of a session.

Definition 7. ***Local Exposure of Session.*** *Let Π be a protocol, and P_i and P_j be two honest parties, $sid_{ij}^{(t)}$ the identifier of a completed session at P_i with peer P_j, and $sid_{ji}^{\prime(t)}$ the matching session's identifier. The session $sid_{ij}^{(t)}$ is said to be locally exposed to an adversary $\mathcal{A}$ if only $\mathcal{A}$ issued a $SK\_Reveal(sid_{ij}^{(t)})$.*

Definition 8. ***Exposure of Session.*** *A session $sid_{ij}^{(t)}$ is said to be exposed if (a) it is locally exposed, or (b) its matching session $sid_{ji}^{\prime(t)}$ exists and is locally exposed.*

Definition 9 ***(Session Freshness).*** *A session that is not exposed is called a fresh session.*

Remark 1. Note that the adversary can ask the following combinations of queries:

1. $LTK\_Reveal\,(i, j, t)$ where $1 \leq l < t$ and $Ephemeral\_Reveal(sid_{ij}^{(t)})$.
2. $LTK\_Reveal\,(i, j, t)$ where $1 \leq l < t$ and $Ephemeral\_Reveal(sid_{ji}^{\prime(t)})$.

The goal of the adversary is to guess whether the challenge is a true session key or a randomly selected key. We say the adversary is successful if he manages to distinguish the session key from a random value with a noticeable probability bounded away from $\frac{1}{2}$ in any non-obvious way. More formally let us define the security of a symmetric key AKE protocol as in Definition 10.

Definition 10 ***(Sym-AKE security).*** *The protocol Π is said to be Sym-AKE-secure, if no polynomially bounded adversary can distinguish a fresh session key from a random value, chosen from the distribution of session keys, with probability significantly greater than 1/2. An adversary $\mathcal{A}$ outputs his guess b' in the test session. The adversary wins the game if he guesses the challenge b correctly, i.e., $b' = b$. The advantage of $\mathcal{A}$ against Π in the Sym-AKE model is defined as*

$$Adv_{\mathcal{A}}^{\Pi,\textbf{Sym-AKE}} = \Pr[b' = b] - \tfrac{1}{2}.$$

The protocol Π is defined to be Sym-AKE secure iff the following two conditions hold:

1. *If two honest parties complete matching sessions, then, except with negligible probability, they both compute the same session key. (Correctness)*
2. *For any probabilistic polynomial-time adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{\Pi,\textbf{Sym-AKE}}$ is negligible.*

4 Symmetric Key Exchange Protocol Resilient to Fully Active (FA) Adversaries

In this section, we give our construction of a symmetric key exchange protocol that can handle fully active adversaries. We give the adversary the power to additionally launch active attacks on the test session. Note that it is trivial to achieve security against FA adversaries in *one-round* by simply MAC-ing or applying a secure pseudo-random function (PRF) on the values sent across by the parties. More precisely party P_i can simply perform a MAC on the value χ_i by using the LTKs shared between P_i and P_j (keyed MAC), i.e., let $y = MAC_K(\chi_i, P_i, P_j)$. Now party P_i sends $\langle P_i, \chi_i, y\rangle$ to party P_j who can verify the authenticity of the MAC. After this party P_j can simply perform a MAC on the value χ_j by using the LTKs shared between P_j and P_i, i.e., let $y' = MAC_K(\chi_j, P_j, P_i)$. Party P_j then sends $\langle P_j, \chi_j, y'\rangle$ to party P_i. Party P_i can verify the authenticity of the MAC sent from P_j and engage in session key exchange. Another way of achieving security against FA adversaries is by using a PRF with the seed of the PRF being the LTK shared between P_i and P_j.

But instead of using MAC or PRF we use weaker primitives like Weak Pseudo Random Functions (wPRF) and weak Almost XOR Universal (wAXU) hash functions to achieve resilience against FA adversaries. We propose a three-pass symmetric key AKE resilient to FA adversaries using concrete instances of these weaker primitives.

4.1 Intuition Behind our Construction

NAXOS trick [LLM07] is one of the standard tool that is used to construct secure key exchange protocols in public key paradigm. We, for the first time, introduce the NAXOS trick in symmetric key setting. We generate a pseudo-ephemeral key $\tilde{K}$ by hashing the long term shared secret key between the parties (LTK) and the ephemeral randomness chosen by the parties in each session ($rand_k$ where $k \in i, j$), i.e., $\tilde{K} \leftarrow H_1(LTK, rand_k)$. The value $\tilde{K}$ is never stored, and so the adversary must learn both LTK and $rand_k$ to compute $\tilde{K}$. So the adversary needs to query LTK_Reveal on one of the two parties and also $Ephemeral\_Reveal$ on any one of the two parties in the current epoch, but this type of query is not allowed in our model. So in our protocol, the initiator of the session must compute $\tilde{K} = H(LTK, rand)$ twice: once during sending $g^{\tilde{K}}$ and once during computing the secret session key from the received values. This is done to avoid to storing this pseudo ephemeral value, which when compromised can allow the adversary to compute the shared secret session key. But in addition to safeguard against active attacks in the test session we use wPRF and wAXU hash functions to achieve authentication of parties. However to achieve a three-pass symmetric key AKE using these primitives in *three-pass* is challenging. The main idea we employ is a trick what we call the *input swapping* technique. Using this trick we get a three-pass symmetric key exchange protocol with mutual authentication of parties in the presence of FA adversaries. The idea is to swap

the inputs of the wPRF and the wAXU hash functions during the computation of values at the two parties as shown in Algorithm 1. More precisely, a party computes the function sampled randomly from the wPRF family on it's local input and evaluates the hash function sampled randomly from the wAXU on the input received from the other party. Now the adversary receives the values sent by one party (owner) and also the tag value computed by its peer (responder) in the second pass. Now he has to forge the tag value of the owner. The tag values computed by the owner and responder have their inputs swapped in the wPRF and the wAXU functions. So the security intuition is that even receiving the tag value from the receiver in the second pass and the values sent by the owner in the first pass will not help the adversary to forge the tag value sent from the owner in the third pass. Also if the adversary tries to tamper with the values sent by the owner or the receiver or both he will be caught during verification since the adversary does not possess the long term secret keys of the current epoch held by the parties.

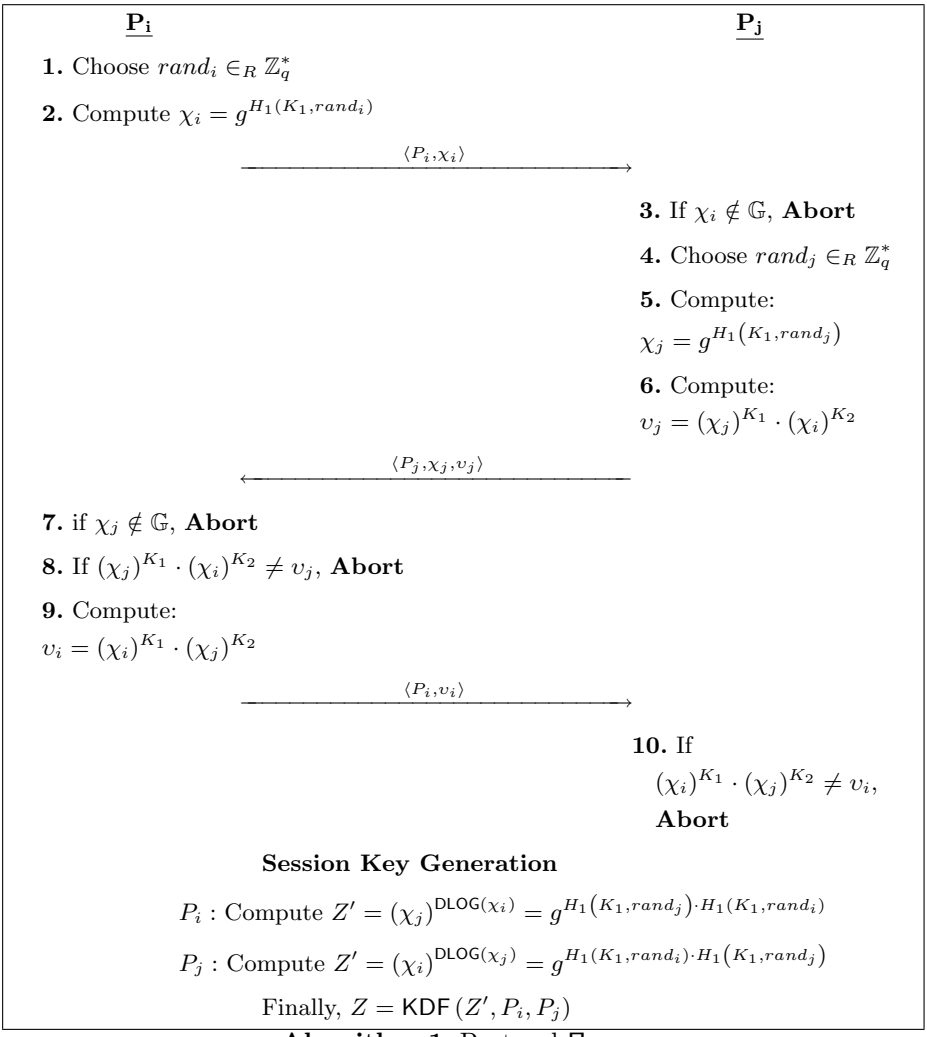
4.2 Protocol Π: A Three-Pass Protocol Secure Against FA Adversaries

In this section we present the concrete construction of a protocol that is secure against fully active adversaries in our new security model. Let P_i and P_j be the two parties participating in the protocol. Our construction provides mutual authentication of both the parties involved in the session key establishment and in addition are *forward-secure*.

Setup: Choose a group $\mathbb{G}$ of prime order q and let g be a generator of $\mathbb{G}$. We assume the DDH problem is hard in $\mathbb{G}$. Let the keyspace $\mathcal{K}_1 = \mathcal{K}_2 = \mathbb{Z}_q^*$. Let $\mathcal{F} = \{f_{K_1} : \mathbb{G} \rightarrow \mathbb{G}\}_{K_1 \in \mathcal{K}_1}$ be a weak pseudo random function(wPRF) family and $\mathcal{H} = \{h_{K_2} : \mathbb{G} \rightarrow \mathbb{G}\}_{K_2 \in \mathcal{K}_2}$ be a weak Almost XOR-Universal (wAXU) hash function family. We may define $f_{K_1}(x) = x^{K_1}$ and $h_{K_2}(y) = y^{K_2}$ so that f is a wPRF under the DDH assumption and h is a also a wAXU hash function by DDH as well [DKPW12]. Choose a collision resistant hash function $H_1 : \mathbb{Z}_p^* \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$ and a non-invertible collision resistant *key derivation function* $\mathsf{KDF} : \mathbb{Z}_q^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_q^*$.

Long Term KeyGen: For every pair of users, say P_i and P_j the Long Term KeyGen algorithm produces a long term key $\text{LTK} = K = \langle K_1, K_2 \rangle \in \mathcal{K}_1 \times \mathcal{K}_2$ and we assume that this common key is securely available with both the parties P_i and P_j.

Session Key Establishment: The two users P_i and P_j choose secret ephemeral exponents $rand_i$ and $rand_j$ respectively and compute the values χ_i and χ_j respectively. They now engage in an interactive three-pass key establishment protocol as shown in Algorithm 1. Party P_j computes a tag υ_j using the keyed function f chosen uniformly at random from the wPRF family and a keyed hash function h uniformly at random from the wAXU family. The party P_j applies the function f on its local input (χ_j) and the function h on the input received from



Algorithm 1: Protocol Π

P_i (i.e., χ_i). Party P_j then send the values χ_j and the computed tag value υ_j to P_i. Party P_i then verifies the tag using the secret key $K = \langle K_1, K_2 \rangle$ and the received value χ_j. If the verification goes through, P_i computes the tag value υ_i and sends it over to P_j and proceed with the session key generation phase. Notice that here we are also using our technique of *input swapping*, i.e., the inputs to the functions f_{K_1} and h_{K_2} are swapped in the computation of υ_j and υ_i). P_j then verifies the received tag value from P_i using the LTK $K = \langle K_1, K_2 \rangle$. If the verification goes through, it proceeds with the session key generation phase. Finally at the end of the protocol both the users P_i and P_j compute the same session key Z' as shown and compute the final session key using the secure KDF.

Remark 2. The protocol consists of two sends by the owner of the session and only one send by the responder (peer) per session. Here each party needs to perform four exponentiations, two calls to the wPRF function f and two calls to the wAXU hash function h.

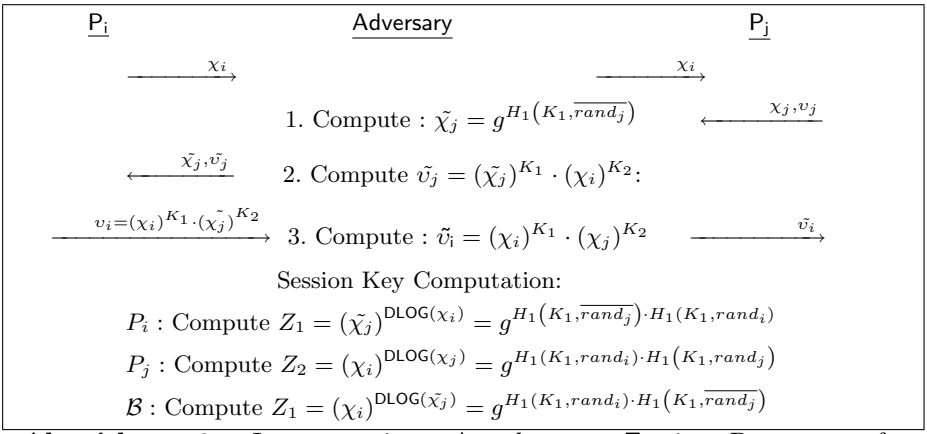
Remark 3. The values χ_k where $k \in \{i, j\}$ are freshly generated for every session. In a preprocessing or a setup stage, the two users generate large number of ephemeral secret values and stores them in the table T_k. For each session, user P_k extracts a fresh value from the table T_k and uses them to generate the session keys for that session.

Remark 4. The value of the LTK $K = \langle K_1, K_2 \rangle$ is updated or *evolved* in each epoch. So the protocol Π is a key-evolving scheme (KES) and also since the values of the ephemeral keys generated in each session are random and independent of each other, it also achieves forward secrecy.

4.3 Need for Disallowing *LTK_Reveal* Query in the Test Session for the Current Epoch

In this section we show that by incorporating active attacks in the test session, it is impossible to achieve any kind of security for the AKE protocol if the long term keys of the parties of the current epoch are given to the adversary. In particular this is not a requirement specific to symmetric key settings only. In public key AKE also the same restriction holds. In particular if we allow $LTK\_Reveal\,(i, j, l)$ oracle service to the adversary for the current epoch T in the test session, then the adversary must be given the LTK K (or private keys in public key settings) shared between parties P_i and P_j for the current epoch. If the adversary $\mathcal{B}$ gets hold of the LTK shared between the two parties and he is allowed to perform active attacks he can successfully launch an *impersonation* attack as shown in Algorithm 2 when he is allowed to query the LTK of P_i and P_j in the current epoch . He can simply chose any session randomness on behalf of any one of the parties and can impersonate that party. Since the adversary also knows the LTK, the authentication at the other end (peer) will go through. This shows an *impossibility* result for allowing the LTK for the current epoch to be revealed to a FA adversary in the test session. The adversary can fool the party P_j into believing that he shares a key with party P_i and he can compute the same session key as party P_i. This represents a successful impersonation attack. In particular we show an impersonation attack on our protocol Π if we allow LTK of current epoch to be given to $\mathcal{B}$. In order to tackle this attack we allow only $LTK\_Reveal\,(i, j, l)$ where $1 \leq l \leq k-1$ and do not reveal the LTK for the current epoch T_k between the users involved in the test session query in our new model. But note that he can even get the LTKs of the users involved in the test session for previous epochs. Hence from the above discussion we get the following proposition:

Proposition 1. *For any key-evolving two-party symmetric key AKE protocol, if we allow the Long Term Key (LTK) of the parties of the current epoch to get revealed to a Fully Active adversary, we cannot hope for any sort of security.*



Algorithm 2: Impersonation Attack on Π in Presence of $LTK\_Reveal(i, j, t)$ where $1 \leq l \leq t$

4.4 Security Proof of Π

In this section we give the security proof of our protocol Π discussed in Sect. 4.2.

Theorem 1. *For any fully active AKE adversary $\mathcal{A}$ against our protocol Π, running in time $\tilde{t}$ and having advantage $\mathsf{Adv}_{\mathcal{A}}^{\Pi, \textbf{\textit{Sym-AKE}}}$, that activates at most m sessions, we construct a DDH solver $\mathcal{S}$ with advantage $\mathsf{Adv}_{\mathcal{S}}^{DDH}$, a discrete logarithm solver τ with advantage $\mathsf{Adv}_{\tau}^{\mathsf{DLOG}}$ such that:*

$$\mathsf{Adv}_{\mathcal{A}}^{\Pi, \textbf{\textit{Sym-AKE}}} \leq \frac{m^2}{2} \cdot \mathsf{Adv}_{\mathcal{S}}^{DDH} - \mathsf{Adv}_{\tau}^{\mathsf{DLOG}} - \frac{1}{2}\left(1 - \frac{1}{|G|}\right).$$

where $\mathcal{S}$ runs in time $O(\tilde{t} \cdot m)$ and τ runs in time $O(\tilde{t})$ and $|G|$ denotes the size of the underlying group.

Proof. Let $\mathcal{A}$ be an adversary against our protocol Π. In accordance with our security model, the adversary $\mathcal{A}$ is allowed to make session activation queries. A query of the form $Send(P_i, P_j)$ makes user P_i perform **Step 1-2** of our protocol, and create a session with identifier $(P_i, P_j, \langle P_i, \chi_i, \eta_i \rangle, \star, \mathcal{I})$. On a query $(P_i, P_j, \langle P_i, \chi_i, \eta_i \rangle)$, P_j creates a session with identifier $(P_j, P_i, \langle P_j, \chi_j, \eta_j \rangle, \langle P_i, \chi_i, \eta_i \rangle, \mathcal{R})$. The query $(P_i, P_j, \langle P_i, \chi_i, \eta_i \rangle, \langle P_j, \chi_j, \eta_j \rangle, \mathcal{R})$ makes P_i update the session identifier $(P_i, P_j, \langle P_i, \chi_i, \eta_i \rangle, \star, \mathcal{I})$ (if any) to $(P_i, P_j, \langle P_i, \chi_i, \eta_i \rangle, \langle P_j, \chi_j, \eta_j \rangle, \mathcal{I})$ and perform **Step 4-6** of our protocol. The adversary is also allowed to make the following queries: *Ephemeral_Reveal*, *LTK_Reveal*, *SK_Reveal*. Since the session key is computed as $H_2(\sigma)$ where $\sigma = (Z', P_i, P_j)$, the adversary can distinguish a fresh session key from a random session key in two ways:

- **Guessing Attack:** $\mathcal{A}$ guesses the test session key correctly.
- **Key Replication Attack:** $\mathcal{A}$ succeeds in making two non-matching sessions compute the same session key and then $\mathcal{A}$ simply issues a session key reveal query on one of the sessions and uses that key in the other session.

– **Forging Attack:** $\mathcal{A}$ computes the value σ and issues the H_2 digest query to get the session key.

Under the Random Oracle (RO) model, the first two attacks cannot succeed, except with negligible probability. Key Replication attack will not succeed, because if $\chi_i \neq \chi'_i$, or $\chi_j \neq \chi'_j$, or $\eta_i \neq \eta'_i$ or $\eta_j \neq \eta'_j$ or $P_i \neq P'_i$ or $P_j \neq P'_j$, and no substring of P_i matches P_j, then the probability that $H_2(\sigma_i, P_i, P_j) = H_2(\sigma'_i, P'_i, P'_j)$ is negligible and vice versa. Thus it is enough to analyze the event E where E is defined as the event "$\mathcal{A}$ succeeds in forging the session key of a fresh session denoted by $sid_0 = (P_i, P_j, \langle \chi_{i_0}, \eta_{i_0}, P_i \rangle, \langle \chi_{j_0}, \eta_{j_0}, P_j \rangle, \varsigma)$". We will show that if $\mathcal{A}$ mounts a successful forgery, then we will be able to construct a DDH solver $\mathcal{S}$ which uses $\mathcal{A}$ as a subroutine.

Analysis of E. If event E occurs with non-negligible probability, using $\mathcal{A}$ we can build a DDH solver that succeeds with non-negligible probability.

$\mathcal{S}$ simulates $\mathcal{A}$'s environment, with n parties $P_1, \ldots, P_n$. Since $\mathcal{A}$ is polynomial (in $|q|$), we suppose that each party is activated at most m times ($m, n \leq \mathcal{L}(|q|$ for some polynomial $\mathcal{L}$). $\mathcal{S}$ chooses P_i, P_j randomly such that $i, j \in [n]$, and $t \in_R [m]$ (with these choices, $\mathcal{S}$ is guessing the test session). The challenger is given the DDH problem instance $\langle \mathbb{G}, g, q, p, C = g^\alpha, D = g^\beta, T \rangle$ where $T = g^{\alpha\beta}$ or T is random.

If the adversary queries on a value σ to the key derivation oracle KDF, the solver $\mathcal{S}$ looks up its corresponding list L_{KDF} to see if the value corresponding to the query is already listed in the list If the KDF *Oracle* was already queried with σ as input, the challenger extracts the value Z from the list L_{KDF} and returns the value. Otherwise it chooses a random value from the distribution of session keys and returns it to the adversary.

The solver $\mathcal{S}$ sets $\chi_{i_0} \leftarrow C$ and $\chi_{j_0} \leftarrow D$. Note that $\mathcal{S}$ does not know the values of $H_1(K, rand_{i0})$ and $H_1(K, rand_{j0})$ and the values of $H_1(K, rand_{i0})$ and $H_1(K, rand_{j0})$ is implicitly set to a and b respectively. We claim that if the adversary wins in the forging attack, $\mathcal{S}$ can solve the DDH challenge. Indeed the session key for the selected test session is of the form $\mathsf{KDF}(\sigma)$ where σ includes the value $CDH(\chi_{i_0}, \chi_{j_0})$, i.e., $CDH(C, D)$. The adversary $\mathcal{A}$ cannot detect whether it is in the simulated session or the actual session unless it queried σ to the KDF function. Hence to win $\mathcal{A}$ must have queried on the KDF at the point σ. So if the adversary guesses the session key correctly, the solver $\mathcal{S}$ outputs 1 indicating it is a DDH tuple, else it outputs 0 indicating it is a random tuple.

Now $\mathcal{A}$ is allowed to make $EphemeralReveal(sid^t_{ij_0})$, $EphemeralReveal(sid'^t_{ji_0})$, but he is not allowed to ask both:

i. $(LTK\_Reveal(i, j, l), EphemeralReveal(sid^t_{ij_0})$ where $l = t$ or
ii. $(LTK\_Reveal(i, j, l), EphemeralReveal(sid^t_{ji_0})$ where $l = t$.

If sid_0 is indeed the test session, the only way $\mathcal{A}$ can distinguish the simulated session from the true session is it makes queries on $(K, rand_{i0})$ or $(K, rand_{j0})$ (by which $\mathcal{A}$ can find out $H_1(K, rand_{i0})$ or $H_1(K, rand_{j0})$). However $\mathcal{A}$ is not allowed to make these queries to reveal both $(K, rand_{i0})$ or $(K, rand_{j0})$. Hence

for this $\mathcal{A}$ needs to find out the discrete logarithms of either $g^{H_1(K,rand_{i0})}$ or $g^{H_1(K,rand_{j0})}$. This corresponds to the hypothetical discrete logarithm solver τ in Theorem 1. Moreover since, $\mathcal{A}$ can activate at most m sessions and make at most $\tilde{t}$ KDF oracle queries, its total running time is $O(\tilde{t} \cdot m)$.

Probability Analysis: The solver $\mathcal{S}$ picks up two parties P_i and P_j and picks a session and its matching session randomly. So the probability that $\mathcal{A}$ picks one of the selected sessions as test session and another as its matching session is $\frac{1}{\binom{m}{2}} = \frac{2}{m(m-1)}$. The advantage of the DDH solver is related to the sum of the advantages of our AKE adversary $\mathcal{A}$ and the discrete logarithm solver τ. Let us define the event E: probability that the adversary $\mathcal{A}$ queries the KDF at the point σ as defined before. Now the advantage of the DDH solver $\mathcal{S}$ is equal to the probability that the adversary $\mathcal{A}$ outputs $b' = b$ (the challenge bit).

$$\Pr[b' = b] = \Pr[b' = b|E] \cdot \Pr[E] + \Pr[b' = b|\neg E] \cdot \Pr[\neg E]$$

Now let us analyze the probability of the event E.

$$\Pr[E] \geq \frac{2}{m(m-1)} \left(\mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathbf{Sym\text{-}AKE}} + \mathsf{Adv}_{\tau}^{\mathsf{DLOG}} \right) \geq \frac{2}{m^2} \left(\mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathbf{Sym\text{-}AKE}} + \mathsf{Adv}_{\tau}^{\mathsf{DLOG}} \right)$$

So, we have:

$$\Pr[b' = b] \geq 1.\frac{2}{m(m-1)} \left(\mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathbf{Sym\text{-}AKE}} + \mathsf{Adv}_{\tau}^{\mathsf{DLOG}} \right) + \frac{1}{2} \cdot \left(1 - \frac{1}{|G|} \right)$$

$$\text{or } \mathsf{Adv}_{\mathcal{S}}^{DDH} \geq 1.\frac{2}{m(m-1)} \left(\mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathbf{Sym\text{-}AKE}} + \mathsf{Adv}_{\tau}^{\mathsf{DLOG}} \right) + \frac{1}{2} \cdot \left(1 - \frac{1}{|G|} \right)$$

$$\text{or } \mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathbf{Sym\text{-}AKE}} \leq \frac{m^2}{2} \cdot \mathsf{Adv}_{\mathcal{S}}^{DDH} - \mathsf{Adv}_{\tau}^{\mathsf{DLOG}} - \frac{1}{2} \left(1 - \frac{1}{|G|} \right).$$

By our assumption, both $\mathsf{Adv}_{\mathcal{S}}^{DDH}$ and $\mathsf{Adv}_{\tau}^{\mathsf{DLOG}}$ are negligible. Hence $\mathsf{Adv}_{\mathcal{A}}^{\Pi,\mathbf{Sym\text{-}AKE}}$ is also negligible. □

Thus, Theorem 1 ensures the security of our authenticated symmetric key AKE protocol.

5 Conclusion and Future Work

We propose a new security model for symmetric key AKE. Our model gives much more power to the adversary and also captures security against fully active adversaries, i.e., adversaries who are active in all the sessions including the test session. We also present a concrete construction providing security in our new security model for symmetric key AKE.

We show how to achieve a three-pass symmetric key AKE secure against fully active adversaries without using MACs or pseudorandom functions. Specifically, we use secure instances of weaker primitives like weak PRF and wAXU hash functions. However, achieving a three-pass AKE using wPRFs and wAXU

functions is non-trivial and for this we introduce a novel technique which we call input swapping technique. Our construction is proven secure in the random oracle model under the DDH assumption. We leave open the problem of construction of a symmetric key AKE protocol in our new security model using weaker primitives, resisting fully active adversaries in the standard model.

Acknowledgments. The first two authors sincerely thank Rishiraj Bhattacharyya for a few technical discussions during the early stage of this work, that clarified some doubts on this topic. Part of this work was done while the first author was visiting R. C. Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata during the Summer of 2015, and the third author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467. The second author is also grateful to the Project CoEC (Centre of Excellence in Cryptology), Indian Statistical Institute, Kolkata, funded by the Government of India, for partial support towards this project.

References

[BCM13] Basin, D., Cremers, C., Meier, S.: Provably repairing the iso/iec 9798 standard for entity authentication. J. Comput. Secur. **21**(6), 817–846 (2013)

[BGH+92] Bird, R.S., Gopal, I., Herzberg, A., Janson, P., Kutten, S., Molva, R., Yung, M.: Systematic design of two-party authentication protocols. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 44–61. Springer, Heidelberg (1992)

[BLL07] Byun, J.K., Lee, D.H., Lim, J.I.: Ec2c-paka: An efficient client-to-client password-authenticated key agreement. Inf. Sci. **177**(19), 3995–4013 (2007)

[Boy90] Boyd, C.: Hidden assumptions in cryptographic protocols. IEE Proc. E (Comput. Digital Tech.) **137**(6), 433–436 (1990)

[BR94] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)

[Chi07] Chien, H.-Y.: Sasi: A new ultralightweight rfid authentication protocol providing strong authentication and strong integrity. IEEE Trans. Dependable Secure Comput. **4**(4), 337–340 (2007)

[CJ97] Clark, J.A., Jacob, J.L.: A survey of authentication protocol literature: Version 1.0. (1997)

[CK01] Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)

[CL08] Cao, T., Lei, H.: Privacy-enhancing authenticated key agreement protocols based on elliptic curve cryptosystem. Acta Electronica Sinica **36**(2), 397 (2008)

[CY08] Cheng, H., Yang, G.: Ekaes: An efficient key agreement and encryption scheme for wireless sensor networks. J. Electron. (China) **25**(4), 495–502 (2008)

[DH76] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)

[DKPW12] Dodis, Y., Kiltz, E., Pietrzak, K., Wichs, D.: Message authentication, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 355–374. Springer, Heidelberg (2012)

[DVOW92] Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Crypt. **2**(2), 107–125 (1992)

[GNY90] Gong, L., Needham, R., Yahalom, R.: Reasoning about belief in cryptographic protocols. In: Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 234–248. IEEE (1990)

[Gün90] Günther, C.G.: An identity-based key-exchange protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)

[JKL04] Jeong, I.R., Katz, J., Lee, D.-H.: One-round protocols for two-party authenticated key exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 220–232. Springer, Heidelberg (2004)

[Kra05] Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)

[LCZ07] Rongxing, L., Cao, Z., Zhu, H.: An enhanced authenticated key agreement protocol for wireless mobile communication. Comput. Stand. Interfaces **29**(6), 647–652 (2007)

[LLM07] LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)

[LMQ+03] Law, L., Menezes, A., Minghua, Q., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Des. Codes Crypt. **28**(2), 119–134 (2003)

[MU08] Menezes, A., Ustaoglu, B.: Comparing the pre- and post-specified peer models for key agreement. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 53–68. Springer, Heidelberg (2008)

[NS78] Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12), 993–999 (1978)

[OR87] Otway, D., Rees, O.: Efficient and timely mutual authentication. ACM SIGOPS Operating Syst. Rev. **21**(1), 8–10 (1987)

[Sat90] Satyanarayanan, M.: Scalable, secure, and highly available distributed file access. Computer **23**(5), 9–18 (1990)

[SEVB10] Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A new security model for authenticated key agreement. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 219–234. Springer, Heidelberg (2010)

[Sho99] Shoup, V.: On formal models for secure key exchange. Citeseer (1999)

[Ust08] Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (h) mqv and naxos. Des. Codes Crypt. **46**(3), 329–342 (2008)