SERVER-ASSISTED LONG-TERM SECURE 3-PARTY KEY ESTABLISHMENT

Kashi Neupane and Rainer Steinwandt
Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, FL, U.S.A.

Keywords: Server-assisted key establishment, Longterm security.

Abstract: Consider a scenario where a server S shares a symmetric key k_U with each user U. Building on a 2-party solution of Bohli et al., we describe an authenticated 3-party key establishment which remains secure if a computational Bilinear Diffie Hellman problem is hard or the server is uncorrupted. If the BDH assumption holds during a protocol execution, but is invalidated later, entity authentication and integrity of the protocol are still guaranteed.

1 INTRODUCTION

In the design of key establishment protocols it is common practice to make use of asymmetric building blocks. A question naturally arising here, especially when aiming at longterm guarantees, is the effect of a violation of an underlying hardness assumption—for instance a discrete logarithm computation might become feasible a few years after a key establishment protocol has been executed. In a server-assisted setting, trying to integrate symmetric techniques as a fall-back technique appears to be a natural approach, and (Bohli et al., 2007a) propose a 3-round protocol for two-party key establishment addressing this scenario: their proposal builds on a symmetric encryption scheme which is secure in a sense reminiscent of left-or-right indistinguishability. Given such a primitive, Bohli et al.'s construction ensures semantic security of the session key if the server is uncorrupted or a Decision Diffie Hellman assumption holds.

Our contribution. The 3-party protocol in the random oracle model presented below enables the establishment of a common session key among 3 parties within 3 rounds. The protocol builds on the Bilinear Diffie Hellman (BDH) assumption and a symmetric encryption scheme which is secure in the sense of real-or-random indistinguishability. Provided that at least one of these two hardness assumptions holds, semantic security of the session key is ensured. In case the BDH assumption is broken *after* completion of the protocol, entity authentication and integrity are still preserved. We did not make an attempt to avoid the random oracle model, but tried to avoid the introduction of new hardness assumptions respectively requirements on the underlying symmetric encryption scheme.

2 PRELIMINARIES

On the mathematical side, the main technical tool is a bilinear pairing. Following the formalization of (Boneh and Franklin, 2003), in the next section we quickly review the relevant terminology—for more details we refer to (Boneh and Franklin, 2003). Similarly, in Section 2.2, we review the main idea of real-or-random indistinguishability as discussed in (Bellare et al., 2000a), and we refer to the latter for a more detailed discussion.

2.1 Bilinear Maps and the Bilinear Diffie Hellman Assumption

Let G_1 and G_2 be two groups of prime order q, such that $q > 2^\ell$ with the security parameter being ℓ. We use additive notation for G_1, multiplicative notation for G_2, and denote by $\hat{e} : G_1 \longrightarrow G_2$ an *admissible bilinear map*, i. e., $\hat{e}$ has all of the following properties:

Bilinear. For all $P, Q \in G_1$ and all $a, b \in \mathbb{Z}$ we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.

Non-degenerate. For $P \neq \mathcal{O}$, we have $\hat{e}(P, P) \neq 1$, i. e., $\hat{e}(P, P)$ is a generator of G_2.

Efficiently Computable. There is a polynomial time algorithm which for all $Q, R \in G_1$ computes $\hat{e}(Q, R)$.

Neupane K. and Steinwandt R. (2010).
SERVER-ASSISTED LONG-TERM SECURE 3-PARTY KEY ESTABLISHMENT.
In *Proceedings of the International Conference on Security and Cryptography*, pages 372-378
DOI: 10.5220/0002983503720378

To specify the Bilinear Diffie-Hellman (BDH) problem, we use a probabilistic polynomial time (ppt) algorithm $\mathcal{G}$: on input the security parameter 1^ℓ, this *BDH parameter generator* $\mathcal{G}$ outputs q and a description of G_1, G_2, and $\hat{e}$ as above; in slight abuse of notation we write $\langle q, G_1, G_2, \hat{e}\rangle \leftarrow \mathcal{G}(1^\ell)$. Descriptions output by $\mathcal{G}$ are assumed to specify polynomial time algorithms for efficiently computing in G_1, G_2 and for evaluating the bilinear map $\hat{e}$.

Next, for a ppt algorithm $\mathcal{A}$ we consider the following experiment:

1. The BDH parameter generator is run, yielding BDH parameters
$$\langle q, G_1, G_2, \hat{e}\rangle.$$
2. Values $a,b,c \leftarrow \{0,\ldots,q-1\}$ are chosen uniformly at random, and $\mathcal{A}$ obtains the output of $\mathcal{G}$ along with aP, bP and cP as input.
3. Now $\mathcal{A}$ outputs a value $g \in G_2$, and is successful whenever $g = \hat{e}(P,P)^{abc}$.

To measure the *advantage of $\mathcal{A}$ in solving the BDH problem* we use the function $\mathrm{Adv}^{\mathsf{bdh}}_{\mathcal{A}} = \mathrm{Adv}^{\mathsf{bdh}}_{\mathcal{G},\mathcal{A}}(\ell) :=$

$$\Pr\left[\mathcal{A}(q, G_1, G_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P,P)^{abc} \;\middle|\; \begin{array}{l}\langle q, G_1, G_2, \hat{e}\rangle \leftarrow \mathcal{G}(1^\ell),\\ P \leftarrow G_1 \setminus \{O\},\\ a,b,c \leftarrow \{0,\ldots,q-1\}\end{array}\right]$$

Definition 1 (BDH Assumption). *A BDH instance generator $\mathcal{G}$ satisfies the BDH assumption if for all ppt algorithms $\mathcal{A}$, the advantage* $\mathrm{Adv}^{\mathsf{bdh}}_{\mathcal{A}}$ *is negligible (in ℓ). In this case, we say that* BDH is hard *in groups generated by $\mathcal{G}$.*

2.2 Real-or-random Indistinguishability

Our presentation of real-or-random indistinguishability follows the one in (Bellare et al., 2000a), and we refer to the latter paper for a more detailed discussion. By a *symmetric encryption scheme*, we mean a collection $\mathcal{SE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of three polynomial time algorithms:

Gen: a probabilistic algorithm that on input the security parameter 1^ℓ outputs a secret key $k \in \{0,1\}^*$;

Enc: a probabilistic algorithm that on input a secret key k and a plaintext $m \in \{0,1\}^*$ outputs a ciphetext $c \in \{0,1\}^*$;

Dec: a deterministic algorithm that on input a secret key k and a ciphertext c outputs the corresponding plaintext m or an error symbol $\perp$. For a valid secret key k output by Gen, we impose that $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$ for all plaintexts $m \in \{0,1\}^*$.

To formalize the security notion needed later, we use a *real-or-random oracle* $\mathcal{E}_k(\mathcal{RR}(\cdot,b))$ that on input $b \in \{0,1\}$ and a plaintext $m \in \{0,1\}^*$ returns an encryption $c \leftarrow \mathsf{Enc}_k(m)$ of m, if $b = 1$. For $b = 0$, an encryption $c \leftarrow \mathsf{Enc}_k(r)$ of a uniformly at random chosen bitstring $r \leftarrow \{0,1\}^{|m|}$ is returned, where $|m|$ denotes the length of m.

For a ppt algorithm $\mathcal{A}$ now consider the following experiment where $b \in \{0,1\}$ is fixed and unknown to $\mathcal{A}$: a secret key $k \leftarrow \mathsf{Gen}(1^\ell)$ is created, and $\mathcal{A}$ has unrestricted access to $\mathcal{E}_k(\mathcal{RR}(\cdot,b))$. Further, $\mathcal{A}$ has access to a decryption oracle $\mathcal{D}_k(\cdot)$ which executes $\mathsf{Dec}_k(\cdot)$, subject to the restriction that no messages must be queried to $\mathcal{D}_k(\cdot)$ that have been output by the real-or-random oracle. We measure $\mathcal{A}$'s advantage as the difference $\mathrm{Adv}^{\mathsf{ror-cca}}_{\mathcal{A}} = \mathrm{Adv}^{\mathsf{ror-cca}}_{\mathcal{A}}(\ell) :=$

$$\Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{E}_k(\mathcal{RR}(\cdot,1)),\mathcal{D}_k(\cdot)}(1^\ell) \,\middle|\, k \leftarrow \mathsf{Gen}(1^\ell)\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{E}_k(\mathcal{RR}(\cdot,0)),\mathcal{D}_k(\cdot)}(1^\ell) \,\middle|\, k \leftarrow \mathsf{Gen}(1^\ell)\right]$$

Definition 2 (Real-or-random Indistinguishability). *A symmetric encryption scheme $\mathcal{SE}$ is* secure in the sense of real-or-random indistinguishability (ROR-CCA)*, if for all ppt algorithms $\mathcal{A}$, the advantage* $\mathrm{Adv}^{\mathsf{ror-cca}}_{\mathcal{A}}$ *is negligible (in ℓ).*

3 SECURITY MODEL

To analyze the security of the proposed protocol, we use a model based on the framework in (Bresson et al., 2001), which in turn is derived from (Bellare et al., 2000b). The latter paper by Bellare et al. also gives more details on the variables that are used below to describe protocol instances.

Protocol Participants. We denote by $U_0 = S$ a dedicated *server* and by $\mathcal{U} = \{U_1, \ldots, U_n\}$ a polynomial size set of *users*.[1] Both server and users are modeled as ppt algorithms, and each $U \in \mathcal{U} \cup \{S\}$ can execute a polynomial number of protocol instances Π^s_U concurrently ($s \in \mathbb{N}$). To describe a protocol instance Π^s_U, seven variables are associated with it:

acc^s_U**:** indicates if the session key stored in sk^s_U has been accepted;

pid^s_U**:** stores the identities of those users in $\mathcal{U}$ with which a key is to be established (including U);

sid^s_U**:** stores a session identifier that can serve as public identifier for the session key stored in sk^s_U;

[1] We assume user identities to be encoded as bitstrings of identical length.

sk_U^s: stores the session key and is initialized with a distinguished NULL value;

state_U^s: stores state information;

term_U^s: indicates if this protocol execution has terminated;

used_U^s: indicates if this instance is used, i. e., involved in a protocol run.

Initialization. Before the actual protocol executions, an initialization phase without adversarial interference takes place. In this phase, for each user $U \in \mathcal{U}$ a verification key/signing key pair $(pk_U, sk_U^{\mathsf{sig}})$ for an existentially unforgeable (UF-CMA secure) signature scheme is generated, sk_U^{sig} is handed to U, and each user U obtains the public keys $pk_{U'}$ for all $U' \in \mathcal{U}$. We denote the signing resp. verification algorithm with Sig resp. Ver. In addition, for each user $U \in \mathcal{U}$, a secret key $k_U \leftarrow \mathsf{Gen}(1^\ell)$ for the underlying symmetric encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is generated; this key is given to U and the server S. Thus, after this initialization phase, the server shares a symmetric key k_U with each user $U \in \mathcal{U}$.

Communication Network and Adversarial Capabilities. The network is non-private, fully asynchronous, and allows arbitrary point-to-point connections among the users and between users and the server. The adversary $\mathcal{A}$ is modeled as ppt algorithm with complete control over the communication network. The following three *oracles* materialize the adversary's capabilities:

$\mathsf{Send}(U_i, s_i, M)$: sends the message M to instance $\Pi_{U_i}^{s_i}$ of user U_i and returns the protocol message output by that instance after receiving M. In addition, the Send oracle is used to initialize a protocol run: to initialize a protocol run of U_i with $U_j, U_k \in \mathcal{U}$ and server S, the special message $M = \{U_i, U_j, U_k\}$ is sent to an unused instance $\prod_{U_i}^{s_i}$. After such a query, $\prod_{U_i}^{s_i}$ initializes its $\mathsf{pid}_{U_i}^{s_i}$-value to $\{U_i, U_j, U_k\}$, sets $\mathsf{used}_{U_i}^{s_i} :=$ TRUE and processes the first step of the protocol.

$\mathsf{Reveal}(U, s)$: returns the session key sk_U^s if $\mathsf{acc}_U^s =$ TRUE and a NULL value otherwise.

$\mathsf{Corrupt}(U)$: for a user $U \in \mathcal{U}$ this query returns U's long term signing key sk_U^{sig} as well as the symmetric key k_U shared between U and the server S; for $U = S$, the list of all symmetric keys k_U $(U \in \mathcal{U})$ is returned, along with the information to which user each such key belongs.

In addition, $\mathcal{A}$ has access to a Test oracle, which can be queried only once: the query $\mathsf{Test}(U, s)$ can be made with an instance Π_U^s that has accepted a session key. Then a bit $b \leftarrow \{0,1\}$ is chosen uniformly at random; for $b = 0$, the session key stored in sk_U^s is returned, and for $b = 1$ a uniformly at random chosen element from the space of session keys is returned.

To exclude useless protocols, subsequently we consider only *correct* key establishment protocols, i. e., in the absence of active attacks a common session key is established, along with common session identifier and matching partner identifier. To define what we mean by a secure key establishment protocol, we rely on the following notion of *partnering*.

Definition 3 (Partnering). *Two instances* $\prod_{U_i}^{s_i}$ *and* $\prod_{U_j}^{s_j}$ *are* partnered *if* $\mathsf{sid}_{U_i}^{s_i} = \mathsf{sid}_{U_j}^{s_j}$, $\mathsf{pid}_{U_i}^{s_i} = \mathsf{pid}_{U_j}^{s_j}$ *and* $\mathsf{acc}_{U_i}^{s_i} = \mathsf{acc}_{U_j}^{s_j} =$ TRUE.

Making use of this definition, we can specify what we mean by a *fresh* instance, i. e., an instance the Test oracle can be queried with:

Definition 4 (Freshness). *An instance* $\prod_{U_i}^{s_i}$ *is said to be* fresh *if the adversary neither queried* $\mathsf{Corrupt}(U_j)$ *for some* $U_j \in \mathsf{pid}_{U_i}^{s_i}$, *nor* $\mathsf{Reveal}(U_j, s_j)$ *for an instance* $\prod_{U_j}^{s_j}$ *that is partnered with* $\prod_{U_i}^{s_i}$.

We write $\mathsf{Succ}_{\mathcal{A}}$ for the event that the adversary $\mathcal{A}$ queries Test with a fresh instance and correctly guesses the random bit b used by the Test oracle and refer to

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}} = \mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}}(\ell) := \left| \Pr[\mathsf{Succ}] - \frac{1}{2} \right|$$

as *advantage* of $\mathcal{A}$.

Definition 5 (Semantic Security). *A key establishment protocol is said to be* (semantically) secure, *if* $Adv_{\mathcal{A}}^{\mathsf{ke}} = \mathrm{Adv}_{\mathcal{A}}^{\mathsf{ke}}(\ell)$ *is negligible for all ppt algorithms* $\mathcal{A}$.

Finally, in formalizing *entity authentication* and *integrity*, we follow the definitions in (Bohli et al., 2007b).

Definition 6 (Strong Entity Authentication). *We say that* strong entity authentication *to an instance* $\Pi_{U_i}^{s_i}$ *is provided if* $\mathsf{acc}_{U_i}^{s_i} =$TRUE *and for all uncorrupted* $U_j \in \mathsf{pid}_{U_i}^{s_i}$ *there exists with overwhelming probability an instance* $\Pi_{U_j}^{s_j}$ *with* $\mathsf{sid}_{U_j}^{s_j} = \mathsf{sid}_{U_i}^{s_i}$ *and* $U_i \in \mathsf{pid}_{U_j}^{s_j}$.

Definition 7 (Integrity). *A key establishment protocol fulfills* integrity *if with overwhelming probability for all instances* $\prod_{U_i}^{s_i}$, $\prod_{U_j}^{s_j}$ *of uncorrupted principals the following holds: if* $\mathsf{acc}_{U_i}^{s_i} = \mathsf{acc}_{U_j}^{s_i} =$TRUE *and* $\mathsf{sid}_{U_i}^{s_i} = \mathsf{sid}_{U_j}^{s_j}$, *then* $\mathsf{sk}_{U_i}^{s_i} = \mathsf{sk}_{U_j}^{s_j}$ *and* $\mathsf{pid}_{U_i}^{s_i} = \mathsf{pid}_{U_j}^{s_j}$.

Round 1:

Computation: Each U_i selects $u_i \leftarrow \{0,\ldots,q-1\}$ uniformly at random, and computes u_iP.

Broadcast: Each U_i broadcasts (U_1,U_2,U_3,u_iP)

Round 2:

Computation: The server S selects $k^{\mathsf{srv}} \leftarrow \{0,1\}^\ell$ uniformly at random and for $i=1,2,3$ computes $c_i := \mathsf{Enc}_{k_{U_i}}(U_1,U_2,U_3,u_iP,k^{\mathsf{srv}})$. Each U_i computes $k^{\mathsf{usr}} = H(\hat{e}(P,P)^{u_1u_2u_3})(= H(\hat{e}(u_2P,u_3P)^{u_1}) = H(\hat{e}(u_1P,u_3P)^{u_2}) = H(\hat{e}(u_1P,u_2P)^{u_3}))$.

Broadcast: The server broadcasts $(U_1,U_2,U_3,c_1,c_2,c_3)$.

Check: Each U_i decrypts c_i and checks consistency of the plaintext with the values sent in Round 1.

Round 3:

Computation: Each U_i computes $mk_i := k^{\mathsf{srv}} \parallel k^{\mathsf{usr}} \parallel U_1 \parallel U_2 \parallel U_3$, sets $k_i^{\mathsf{conf}} := H(mk_i \parallel 00)$, and $\sigma_i := \mathsf{Sig}_{sk^{\mathsf{sig}}}(U_1,U_2,U_3,u_1P,u_2P,u_3P,k_i^{\mathsf{conf}})$.

Broadcast: Each U_i broadcasts (U_1,U_2,U_3,σ_i)

Check: Each U_i verifies the signatures σ_j $(j \neq i)$, using the values from Round 1, the value k_i^{conf} just computed, and the public verification keys pk_{U_j}.

Key derivation: If none of the checks failed, U_i sets $\mathsf{sid}_{U_i} := H(mk_i \parallel 01)$, $\mathsf{sk}_{U_i} := H(mk_i \parallel 10)$, and then $\mathsf{acc}_{U_i} :=$TRUE.

Figure 1: Long-term secure 3-party key establishment among users U_1,U_2,U_3, invoking a server S.

4 THE PROPOSED 3-PARTY PROTOCOL

The proposed protocol has three rounds with a total of seven messages being sent, and makes use of a random oracle $H:\{0,1\}^* \to \{0,1\}^\ell$. To describe the protocol we use the notation from Section 2 with P being a generator of the additive group G_1 of prime order q, as used in the BDH assumption. By Enc we denote the encryption algorithm of a symmetric encryption scheme that is secure in the sense of ROR-CCA, and by $\mathcal{S}$ resp. $\mathcal{V}$ we denote the signature resp. verification algorithm of an existentially unforgeable signature scheme. With this notation, the proposed protocol for establishing a common session key among users U_1, U_2, U_3, invoking a server S, is described in Figure 1 (for ease of notation, we omit indices referring to a particular user instance and write only sid_U instead of sid_U^s etc.).

5 SECURITY ANALYSIS

The security of the protocol in Figure 1 can be ensured in the "long-term" provided that the underlying signature scheme is existentially unforgeable and the invoked symmetric encryption scheme is secure in the sense of ROR-CCA. More specifically, we have the following.

Proposition 1. *Suppose the signature scheme used in the protocol in Figure 1 is secure in the sense of* UF-CMA *and the symmetric encryption scheme is secure in the sense of* ROR-CCA*. Then the protocol in Figure 1 is secure, if the invoked signature scheme is existentially unforgeable and at least one of the following conditions holds:*

- *The server S is uncorrupted.*
- *The BDH assumption for the underlying BDH instance generator holds.*

If the above two assumptions hold during the protocol execution (only), then the protocol in Figure 1 still guarantees integrity and strong entity authentication.

Proof. Let q_{send} and q_{ro} be polynomial upper bounds for the number of the adversary $\mathcal{A}$'s queries to the Send oracle and random oracle H, respectively. We begin by defining three events and argue that each of them can occur with negligible probability only:

Forge: this is the event that $\mathcal{A}$ succeeds in forging a signature σ_i of a protocol participant U_i on a Round 3 message without having queried $\mathsf{Corrupt}(U_i)$. Let $\mathrm{Adv}^{\mathsf{uf}} = \mathrm{Adv}^{\mathsf{uf}}(\ell)$ be a negligible upper bound for the probability that a ppt adversary creates a successful forgery for the underlying signature scheme. During the protocol's initialization phase, we can assign a challenge verification key to a user $U \in \mathcal{U}$ uniformly at random, and with probability at least $1/|\mathcal{U}| = 1/n$

the event Forge results in a successful forgery for the challenge verification key. Thus

$$\Pr[\mathsf{Forge}] \leq n \cdot \mathrm{Adv}^{\mathsf{uf}},$$

i. e., Forge can occur with negligible probability only.

Repeat: this is the event where the server S uses the same value k^{srv} more than once, or a user outputs a value u_iP with $u_iP = u_jP$ for a value u_jP that has already been output by some (possibly the same) user earlier. Both k^{srv} and u_iP are only chosen in response to a Send query, and only one such value is created per Send query. Consequently, we have

$$\Pr[\mathsf{Repeat}] \leq \sum_{i=1}^{q_{\mathsf{send}}} \frac{i-1}{2^\ell} \leq q_{\mathsf{send}}^2/2^\ell,$$

i. e., Repeat can occur with negligible probability only.

Collision: this is the event of a collision in the random oracle H, i. e., H produces the same output value for two different input values. As a Send query causes at most two random oracle queries, we can bound the total number of queries to H by $2 \cdot q_{\mathsf{send}} + q_{\mathsf{ro}}$. Therefore

$$\Pr[\mathsf{Collision}] \leq (2 \cdot q_{\mathsf{send}} + q_{\mathsf{ro}})^2/2^\ell$$

is negligible.

As each of the events Forge, Repeat, Collision occurs with negligible probability only, subsequently we may assume they do not occur. Now, for proving security in the sense of Definition 3, game hopping turns out to be convenient. The event of $\mathcal{A}$ to succeed in Game i and the advantage of $\mathcal{A}$ in Game i will be denoted by $\mathsf{Succ}_{\mathcal{A}}^{\text{Game } i}$ and $\mathrm{Adv}_{\mathcal{A}}^{\text{Game } i}$, respectively. First we discuss the situation where the BDH assumption holds; the case of having (only) an uncorrupted server will be discussed thereafter.

Security if the BDH Assumption Holds. A short sequence of games can be used to establish the desired result in this case:

Game 0. This game is identical to the original attack game for the adversary, with all oracles being simulated faithfully. In particular,

$$\mathrm{Adv}_{\mathcal{A}} = \mathrm{Adv}_{\mathcal{A}}^{\text{Game } 0}.$$

Game 1. Here we modify the simulation as follows: In Round 3, if none of the users in pid_i is corrupted, k^{usr} is chosen uniformly at random from $\{0,1\}^\ell$, instead of being computed as $H(\hat{e}(P,P)^{u_1u_2u_3})$.

We claim that $|\mathrm{Adv}_{\mathcal{A}}^{\text{Game } 1} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game } 0}|$ is negligible. To see this, consider the following algorithm $\mathcal{B}$ to solve the BDH problem: On input a BDH challenge with group elements $(P, aP, bP, cP) \in G_1^4$, $\mathcal{B}$ will act as challenger for the adversary $\mathcal{A}$ and choose three protocol instances $\Pi_{U_i}^{s_i}, \Pi_{U_j}^{s_j}, \Pi_{U_k}^{s_k}$ by guessing uniformly at random among all $\leq q_{\mathsf{send}}$ instances queried to the Send oracle.

With probability at least $1/q_{\mathsf{send}}^3$, the adversary $\mathcal{A}$ queries $\mathsf{Test}(U_i, s_i)$ with $\mathsf{pid}_{U_i}^{s_i} = \{U_i, U_j, U_k\}$— in all other cases $\mathcal{B}$ aborts. In Round 1, $\mathcal{B}$ replaces the messages of U_i, U_j, U_k with aP, bP and cP accordingly, and as the Test session must not be revealed, this is unnoticeable to $\mathcal{A}$. Game 1 differs only from Game 0, if $\mathcal{A}$ queries H with $\hat{e}(P,P)^{u_1u_2u_3}$, and whenever $\mathcal{A}$ recognizes the session key correctly, $\mathcal{B}$ chooses one of the $\leq q_{\mathsf{ro}}$ values queried to H uniformly at random and outputs this value as potential solution to the BDH challenge. We obtain

$$\begin{aligned} &\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game } 1} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game } 0}\right| \\ &\leq \left|\Pr[\mathsf{Succ}_{\mathcal{A}}^{\text{Game } 1}] - \Pr[\mathsf{Succ}_{\mathcal{A}}^{\text{Game } 0}]\right| \\ &\leq q_{\mathsf{send}}^3 \cdot q_{\mathsf{ro}} \cdot \mathrm{Adv}_{\mathcal{A}}^{\mathsf{bdh}}, \end{aligned}$$

i. e., $\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game } 1} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game } 0}\right|$ is bounded by a negligible function as desired.

Game 2. Here we replace the session key sk_{U_i} (as well as sk_{U_j} and sk_{U_k}) with a uniformly at random chosen bitstring in $\{0,1\}^\ell$. Game 2 and Game 1 only differ if the adversary queries the random oracle H with a bitstring of the form $* \parallel k^{\mathsf{usr}} \parallel *$. With no information about $k^{\mathsf{usr}} \in \{0,1\}^\ell$ other than $H(mk_i \parallel 00)$ and $H(mk_i \parallel 01)$ being available to $\mathcal{A}$, we obtain

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game } 2} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game } 1}\right| \leq \frac{q_{\mathsf{ro}} + 2 \cdot q_{\mathsf{send}}}{2^\ell}.$$

By construction $\mathrm{Adv}_{\mathcal{A}}^{\text{Game } 2} = 0$, and we recognize the protcol in Figure 1 as secure, provided that the BDH assumption holds.

Security if the Server is Uncorrupted. In other words, $\mathcal{A}$ must not query $\mathsf{Corrupt}(S)$. For this scenario, again game hopping allows to establish the desired result:

Game 0. As in the previous setting, this game is identical to the original attack game for the adversary, with all oracles being simulated faithfully:

$$\mathrm{Adv}_{\mathcal{A}} = \mathrm{Adv}_{\mathcal{A}}^{\text{Game } 0}$$

Game 1. In this game we modify $\mathcal{A}$ in such a way that it chooses first of all, independently and uniformly at random, three protocol instances $\Pi_{U_i}^{s_i}$, $\Pi_{U_j}^{s_j}$, $\Pi_{U_k}^{s_k}$ of the at most $\leq q_{\mathsf{send}}$ instances queried to the Send oracle. With probability at least $1/q_{\mathsf{send}}^3$, the adversary $\mathcal{A}$ will query $\mathsf{Test}(U_i, s_i)$ with $\mathsf{pid}_{U_i}^{s_i} = \{U_i, U_j, U_k\}$—in all other cases just a uniformly at random chosen bit $b \in \{0,1\}$ is output. We have $\mathrm{Adv}_{\mathcal{A}}^{\text{Game 0}} \leq q_{\mathsf{send}}^3 \cdot \mathrm{Adv}_{\mathcal{A}}^{\text{Game 1}}$.

Game 2. Now, in Round 2 of the protocol the simulator replaces the server's message c_i directed to $\Pi_{U_i}^{s_i}$ with an encryption of a uniformly chosen random bitstring of the appropriate length. To bound $|\mathrm{Adv}_{\mathcal{A}}^{\text{Game 2}} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game 1}}|$ we derive from the challenger the following algorithm $\mathcal{C}$ to attack the ROR-CCA security of the underlying symmetric encryption scheme: whenever the protocol requires to encrypt or decypt a message using the symmetric key k_{U_i}, $\mathcal{C}$ queries its encryption or decryption oracle, respectively, simulating Corrupt, Reveal, Send and Test in the obvious way. Note that $\mathcal{C}$ simulates the (by assumption uncorrupted) server S, too. In particular, $\mathcal{C}$ knows k^{srv}, and there is no need for $\mathcal{C}$ to query its decryption oracle with a message received from the real-or-random oracle for computing the session key. Whenever $\mathcal{A}$ correctly identifies the session key after receiving the challenge of the (simulated) Test oracle, $\mathcal{C}$ outpus 1, i. e., claims that its encryption oracle operates in "real mode", whenever $\mathcal{A}$ guesses incorrectly, $\mathcal{C}$ outputs 0.

Writing b^{ror} and b^{test} for the values of the real-or-random oracle's internal random bit and the random bit of the (simulated) test oracle, respectively, we obtain (with a slight abuse of notation) $\left|\mathrm{Adv}_{\mathcal{C}}^{\mathsf{ror-cca}}\right| =$

$$
\begin{aligned}
&\left|\Pr\left[1 \leftarrow \mathcal{C}^{b^{\mathsf{ror}}=1}\right] - \Pr\left[1 \leftarrow \mathcal{C}^{b^{\mathsf{ror}}=0}\right]\right| = \\
&= \Bigg|\frac{1}{2}\cdot\Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=1} \mid b^{\mathsf{ror}}=1\right] \\
&\quad + \frac{1}{2}\cdot\Pr\left[0 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=0} \mid b^{\mathsf{ror}}=1\right] \\
&\quad - \frac{1}{2}\cdot\Pr\left[0 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=1} \mid b^{\mathsf{ror}}=0\right] \\
&\quad - \frac{1}{2}\cdot\Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=0} \mid b^{\mathsf{ror}}=0\right]\Bigg| \\
&= \frac{1}{2}\cdot\Big|\Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=1} \mid b^{\mathsf{ror}}=1\right] \\
&\quad + \left(1 - \Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=0} \mid b^{\mathsf{ror}}=1\right]\right) \\
&\quad - \left(1 - \Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=1} \mid b^{\mathsf{ror}}=0\right]\right) \\
&\quad - \Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=0} \mid b^{\mathsf{ror}}=0\right]\Big| \\
&= \frac{1}{2}\cdot\Big|\Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=1} \mid b^{\mathsf{ror}}=1\right] \\
&\quad - \Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=0} \mid b^{\mathsf{ror}}=1\right] \\
&\quad + \left(\Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=1} \mid b^{\mathsf{ror}}=0\right]\right. \\
&\quad \left. - \Pr\left[1 \leftarrow \mathcal{A}^{b^{\mathsf{test}}=0} \mid b^{\mathsf{ror}}=0\right]\right)\Big| \\
&\geq \frac{1}{2}\cdot\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game 0}} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game 1}}\right|.
\end{aligned}
$$

In other words, we recognize $|\mathrm{Adv}_{\mathcal{A}}^{\text{Game 2}} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game 1}}|$ as negligible as required.

Game 3. In this game, in Round 2 of the protocol the simulator replaces the server's message c_j directed to $\Pi_{U_j}^{s_j}$ with an encryption of a uniformly chosen random bitstring of the appropriate length. With the same argument as above, we recognize $\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game 3}} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game 2}}\right|$ as negligible.

Game 4. Finally, in this game, in Round 2 of the protocol the simulator replaces the server's message c_k directed to $\Pi_{U_k}^{s_k}$ with an encryption of a uniformly chosen random bitstring of the appropriate length. Repeating the argument for Game 2 again, we recognize $\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game 4}} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game 3}}\right|$ as negligible.

Game 5. At this point we replace the session key sk_{U_i} (as well as sk_{U_j} and sk_{U_k}) with a uniformly at random chosen bitstring in $\{0,1\}^\ell$. Game 4 and Game 3 only differ if the adversary queries the random oracle H with a bitstring of the form $k^{\mathsf{srv}} \parallel *$. With no information about $k^{\mathsf{srv}} \in \{0,1\}^\ell$ other than $H(mk_i \parallel 00)$ and $H(mk_i \parallel 01)$ being available to $\mathcal{A}$, we obtain

$$\left|\mathrm{Adv}_{\mathcal{A}}^{\text{Game 4}} - \mathrm{Adv}_{\mathcal{A}}^{\text{Game 5}}\right| \leq \frac{q_{\mathsf{ro}} + 2 \cdot q_{\mathsf{send}}}{2^\ell}.$$

By construction $\mathrm{Adv}_{\mathcal{A}}^{\text{Game 5}} = 0$, and we recognize the protcol in Figure 1 as secure, provided that the server S is uncorrupted.

Integrity. If three instances of honest users agree on a common session identifier $H(mk \parallel 01)$, unless the event Collision occurs they have obtained the same "master key" mk—and therewith partner identifier. With the session key being computed as $H(mk \parallel 10)$, we see that equality of session identifiers with overwhelming probability ensures identical session keys, too.

Strong Entity Authentication. The session identifier is derived from the "master key" mk as $H(mk \parallel$

01), and mk is derived from values u_jP, u_kP, received from and signed by the intended partners; mk also includes the partner identifier. The partner instances know the same values and derived with overwhelming probability an identical confirmation key k^{conf} and therewith an identical session identifier. □

6 CONCLUSIONS

The server assisted 3-party protocol we presented can be seen as expensive in the sense that shared keys with a server, a signature scheme and two hardness assumptions are involved. However, the security guarantee established is rather strong and the efficiency as well as the hardness assumptions compare in our opinion quite acceptably to Bohli et al.'s two-party solution. Avoiding the introduction of new hardness assumptions about the involved cryptographic primitives can certainly be seen as a feature of the presented protocol.

REFERENCES

Bellare, M., Desai, A., Jokipii, E., and Rogaway, P. (2000a). A Concrete Security Treatment of Symmetric Encryption. Available at http://cseweb.ucsd.edu/∼mihir/papers/sym-enc.html. Extended abstract in (Boneh and Franklin, 2001).

Bellare, M., Pointcheval, D., and Rogaway, P. (2000b). Authenticated Key Exchange Secure against Dictionary Attacks. In Preneel, B., editor, *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer.

Bohli, J.-M., Müller-Quade, J., and Röhrich, S. (2007a). *Long-Term and Dynamical Aspects of Information Security: Emerging Trends in Information and Communication Security*, chapter Long-term Secure Key Establishment, pages 87–95. Nova Science Publishers.

Bohli, J.-M., Vasco, M. I. G., and Steinwandt, R. (2007b). Secure group key establishment revisited. *International Journal of Information Security*, 6(4):243–254.

Boneh, D. and Franklin, M. (2001). Identity-Based Encryption from the Weil Pairing. In Kilian, J., editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag.

Boneh, D. and Franklin, M. (2003). Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615. Available at http://crypto.stanford.edu/∼dabo/papers/bfibe.pdf; extended abstract in (Boneh and Franklin, 2001).

Bresson, E., Chevassut, O., Pointcheval, D., and Quisquater, J.-J. (2001). Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security CCS'01*, pages 255–264. ACM.