SCIENCE CHINA
Information Sciences

• **Supplementary File** •

An anonymous key agreement protocol with robust authentication for smart grid infrastructure

Ting CHEN[1], Qingfeng CHENG[2] & Xinghua LI[1*]

[1]*School of Cyber Engineering, Xidian University, Xi'an* 710071*, China;*
[2]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou* 450001*, China*

Appendix A Preliminaries

Since this letter mainly uses bilinear pairing to design scheme and the scheme is based on some known computational difficulty problems, this section will briefly illustrate bilinear pairing and those hard computation problems.

Appendix A.1 Bilinear pairing

Let G_1 be a cyclic additive group and G_2 be a cyclic multiplicative group. Both G_1 and G_2 have the large prime order q. P is assumed to be the generator of G_1. If function e defined as $G1 \times G1 \rightarrow G2$ exists, then (G_1, G_2) is bilinear pairing map groups. Bilinear pairing has some properties described as following.

- Bilinearity: $e(aP, bP) = e(P, P)^{ab}$, where $a, b \in Z_p^*$
- Non-degeneracy: Generator P satisfies $e(P, P) \neq 1$. 1 is G_2's multiplicative identity.
- Computability: An efficient algorithm exists to calculate $e(P, P)$.

Appendix A.2 Problems of computational difficulty

This subsection gives the definitions of elliptic curve discrete logarithm problem (ECDLP), computational Diffie-Hellman problem (CDHP), and divisible computation Diffie-Hellman problem (DCDHP). Descriptions are as below.

- ECDLP: Let P be an elliptic curve point of the prime order q and be the generator of G_1. Given $P, M \in G_1$, it is infeasible to calculate a from $M = aP$, where $a \in Z_p^*$.
- CDHP: Let parameter g be a generator of G_2 and it has the prime order q. Given (g, g^x, g^y), g^{xy} is difficult to be computed , where $x, y \in Z_p^*$.
- DCDHP: Let parameter g be a generator of G_2 and it has the prime order q. Given (g, g^x, g^y), $g^{xy^{-1}}$ is hard to be computed , where $x, y \in Z_p^*$.

Appendix B The explanations of the notions used in our scheme

The notions used in our scheme are explained in Table B1.

Appendix C Security analysis

This section conducts security analysis of the new proposed scheme from following two aspects: formal security verification by tool ProVerif and provable security through random oracle model. The details are described as below.

* Corresponding author (email: xhli1@mail.xidian.edu.cn)

Table B1 The notions and corresponding explanations of our scheme

Notions	Corresponding explanations
TA	The trusted authority
SM_i	The smart meter
UC_j	The utility control
E	The adversary
ID_i	The smart meter SM_i's identity
ID_j	The utility control UC_j's identity
S_i	The private key of the smart meter SM_i
S_j	The private key of the utility control UC_j
s	The master key selected by the trusted authority
P_{pub}	The trusted authority's public key and its value is sP

Appendix C.1 Formal security verification by ProVerif

ProVerif is a tool used for automatic formal verification and analysis of cryptographic protocols, which can verify security attributes such as confidentiality and authentication. Based on Dolev-Yao model, ProVerif uses applied π calculus and verifies protocol's security by queries. Here we present security verification of our protocol by ProVerif tool.

We first define c as the public channel, where the smart meter SM_i and utility control UC_j communicate with each other and exchange information. Additionally, it is important to define the type of parameters, since parameters' types determine whether they can be obtained by adversaries. Public parameters and parameters transmitted over public channel are exposed to adversaries, but private parameters are protected. We define constant P as a public bitstring. Key S_i and S_j, master key s, and session key sk_{ij} are defined as private bitstrings. Afterwards, we need to construct some functions used to describe the proposed protocol. Seven kinds of operations are adopted. The first one is hash functions. The subsequent six operations are multiplication, multiplicative inverse, addition, elliptic curve point multiplication, concatenation, and bilinear pair computation.

After that, we define four events to verify the authentication of the proposed protocol. As far as SM_i is concerned, event $begin-SM_i(bitstring)$ and event $end-SMi$(bitstring) should be defined. From UC_j's viewpoint, event $begin-UC_j(bitstring)$ and event $end-UC_j(bitstring)$ need to be defined. Subsequently, we establish π calculus for the trusted authority TA, the smart meter SM_i, and the utility control UC_j.

Finally, we perform parallel execution of above three processes and make three queries to check the secrecy of the session key sk_{ij} and the authentication between SM_i and UC_j. The results are seen as follows.

$--Query\ not\ attacker(sk_{ij}[])$
$Completing...$
$Starting\ query\ not\ attacker(sk_{ij}[])$
$RESULT\ not\ attacker(sk_{ij}[])\ is\ true.$
$--Query\ inj-event(end-SM_i(ID_j))==>inj-event(begin-SMi(ID_i))$
$Completing...$
$Starting\ query\ inj-event(end-SM_i(ID_j))==>inj-event(begin-SM_i(ID_i))$
$RESULT\ inj-event(end-SM_i(ID_j))==>inj-event(begin-SM_i(ID_i))\ is\ true.$
$--Query\ inj-event(end-UC_j(ID_i))==>inj-event(begin-UC_j(ID_j))$
$Completing...$
$Starting\ query\ inj-event(end-UC_j(ID_i))==>inj-event(begin-UC_j(ID_j))$
$RESULT\ inj-event(end-UC_j(ID_i))==>inj-event(begin-UC_j(ID_j))\ is\ true.$

From the results, we declare that our scheme realizes the secrecy and mutual authentication. Especially, primitive "$RESULT\ not\ attacker(sk_{ij}[])\ is\ true$" means that the session key established between SM_i and UC_j is not capable of being obtained by attackers and it is secret. In addition, primitive "$RESULT\ inj-event(end-SM_i(ID_j))==>inj-event(begin-SM_i(ID_i))\ is\ true$" and primitive "$RESULT\ inj-event(end-UC_j(ID_i))==>inj-event(begin-UC_j(ID_j))\ is\ true$" denote that SM_i verifies the validity of UC_j and UC_j verifies the validity of SM_i.

Appendix C.2 Threat model

This letter adopts the Canetti-Krawczyk (CK) threat model [1]. In this treat model, attackers are Turing machines with probabilistic polynomial time, who are able to completely control the communication channel of participants and can delay, eavesdrop, modify, replay and insert messages arbitrarily. Furthermore, for adversaries in non-authenticated links, they can also initiate a series of queries to obtain the secret information of participants and related sessions. Specifically, an adversary E is permitted to use following queries to communicate with a protocol entity P (or an oracle), where the protocol entity can be either SM_i or UC_j. If an attacker cannot distinguish the session key negotiated by protocol participants with an independent random number under the allowable attack capability, we say that the key agreement protocol satisfies security.

• $Execute(SM_i, UC_j)$: This query simulates E's passive attacks. E is able to obtain all the messages transmitted between SM_i and UC_j by this query.

- $H_k(m)$: This query returns a random number Hu_k to adversary E.
- $Send(P, m)$: This query simulates E's active attacks. After E sends a message m to P, P returns a corresponding response.
- $SSReveal(P)$: This query lets E get the session-specific state information held by P. However, it does not output the long-term private key held by P by this query.
- $SKReveal(P)$: By this query, E is permitted to acquire the session key held by P.
- $Corrupt(P)$: By this query, E is permitted to acquire the long-term private key held by P.
- $Expire(P)$: By this query, E is permitted to delete a completed session's session key held by P.
- $Test(P)$: This query measures the session key's semantic security. In this query, a coin d is flipped by P, where $d = 0, 1$. The real session key is returned by P if $d = 1$. Otherwise, P randomly selects a number and sends it to E.

Some related definitions are as follows.

Definition 1. If oracles SM_i and UC_j authenticate mutually and generate the same sk_{ij}, they are said to be partners.

Definition 2. A session $\prod$ held by oracle P is called locally exposed if queries of $SKReveal(P)$, $SSReveal(P)$, and $Corrupt(P)$ are asked prior to $Expire(P)$ query by adversary E or oracle P is corrupted before the establishment of $\prod$. If both session $\prod$ and its matching sessions are not locally exposed, $\prod$ is considered fresh.

Definition 3. In our proposed authenticated key agreement (AKA) scheme, the security is simulated by game $GM^{AKA}(P, E)$, where E is permitted to issue many $Test(P)$ (SM_i or UC_j) queries. After receiving a $Test(P)$ query, P flips a coin d and responds with d, where d is either an actual session key or a random number. Adversary E guesses a bit $d^{'}$. If $d^{'}$=d, E wins the game. Let $Pr[d^{'}=d]$ be the probability that E successfully wins $GM^{AKA}(P, E)$, then E 's advantage in violating our scheme's semantic security is denoted as $Adv_{AKA}(E)$=$|2Pr[d^{'}=d] - 1|$.

Definition 4. If $Adv_{AKA}(E) \leqslant \varepsilon$ holds, where ε is small enough, then the new proposed scheme is said to resist CK-adversary model.

Appendix C.3 Provable security through random oracle model

Subsequently, we proof our scheme's security under random oracle model. We first present difference lemma [2] necessary for analysis and give the theorem which needs to be proofed.

Lemma 1 (Difference Lemma)**.** Let B_1, B_2, B_3 be the events defined in some probability distribution. Given $B_1 \wedge \neg B_3 \Leftrightarrow B_2 \wedge \neg B_3$, we can conclude $|Pr[B_1] - Pr[B_2]| \leqslant Pr[B_3]$.

Theorem 1. Suppose an adversary E is probabilistic polynomial time-bounded, who can break our scheme's semantic security by issuing $Hash$ query for at most q_h times, $Execute$ query for at most q_e times and $Send$ query for at most q_s times. E's advantage is defined as follows.

$$Adv_{AKA}(E) \leqslant \frac{4q_h^2 + 2q_s^2}{2^l} + \frac{(q_s + q_e)^2}{p} + 2q_h Adv_{CDHP}(E)$$

Proof. We define six games $GM_0 \sim GM_5$ to prove our scheme's semantic security. Especially, GM_0 denotes the actual attack and adversary E has no advantage in GM_5. We assume that Suc_j is GM_j's corresponding event, which denotes E's success guessing of bit d by $Test$ query.

Game GM_0: In random oracle model, this game represents real attack. Thus, we can obtain following conclusion.

$$Adv_{AKA}(E) = |2Pr[Suc_0] - 1| \tag{C1}$$

Game GM_1: This game simulates all the oracles and the details are shown in Table C1. The answers are kept in corresponding lists. As the simulation shows, we can observe that GM_1 is indistinguishable from GM_0. Thus, we can obtain following conclusion.

$$Pr[Suc_1] = Pr[Suc_0] \tag{C2}$$

Game GM_2: This game nearly simulates GM_1. The difference is that GM_2 ignores several collisions of M_1 or M_2 in the hash queries and transcripts, where M_1 and M_2 are random. Thus, this game and the previous game are indistinguishable if there are no collisions occur. By applying birthday paradox, hash collisions' probability is no more than $\frac{q_h^2}{2^l}$. Similarly, in transcripts, the collisions' probability is no more than $\frac{(q_s+q_e)^2}{2p}$. As a result, by adopting Lemma 1, we can get following conclusion.

$$|Pr[Suc_2] - Pr[Suc_1]| \leqslant \frac{q_h^2}{2^l} + \frac{(q_s + q_e)^2}{2p} \tag{C3}$$

Game GM_3: This game is identical to GM_2 unless SM_i refuses Y_2 or UC_j refuses Y_1 or G_i. Moreover, GM_3 will be terminated if adversary E is fortunate to guess the values of Y_1, Y_2, and G_i without inquiring the H oracle. Thus, we can obtain following conclusion.

$$|Pr[Suc_3] - Pr[Suc_2]| \leqslant \frac{q_s^2}{2^l} \tag{C4}$$

Game GM_4: This game considers the security of the session key. This security property guarantees that E still fails to acquire sk_{ij} unless she obtains SM_i's secret information $\{a, x_1, S_i\}$ or UC_j's secrecy $\{b, x_2, S_j\}$. The goal of E is to calculate sk_{ij} in the following cases by issuing some $Hash$ and $Execute$ queries, who is permitted to obtain session transcripts.

Table C1 Simulation of oracles

For a one way hash query $H_k(m)$, where $k = 1, ..., 5$, if there is a record (Hu_k, m) exists in the list L_{Hu_k}, then Hu_k is returned. Otherwise, a random number Hu_k is selected from $\{0,1\}^l$ and is returned to adversary E. After that, the new record Hu_k is added into L_{Hu_k}.
For a query $Send(SM_i, begin)$, oracle SM_i randomly selects a, x_1 from Z_p^*, computes $Z{=}g^{a+H_3(S_i\|x_1)}$, $A_j{=}P_{pub}{+}H_1(ID_i)P$, $M_1{=}(a + H_3(S_i\|x_1)) \cdot A_j$, and $Y_1 {=}H_4(ID_i\|ID_j\|Z\|M_1)$, and responds with $\{M_1, Y_1\}$.
For a query $Send(UC_j, \{M_1, Y_1\})$, oracle UC_j chooses two random numbers b, x_2 from Z_p^*, computes $Z = e(M_1, S_j){=}e(P, P)^{a+H_3(S_i\|x_1)}$, and then verifies the correctness of $Y_1 {=}H_4(ID_i\|ID_j\|Z\|M_1)$. If it is valid, UC_j determines $V = e(P, P)^{b+H_3(S_j\|x_2)} = g^{b+H_3(S_j\|x_2)}$, further generates $K_{ij}{=}H_2(Z^{b+H_3(S_j\|x_2)})$, $sk_{ij} = H_4(ID_i\|ID_j\|Z\|V\|K_{ij})$, $B_i = P_{pub} + H_1(ID_i)P$, $M_2 = (b + H_3(S_j\|x_2))\cdot B_i$, and $Y_2 = H_4(ID_i\|ID_j\|V\|M_2)$, and finally replies with $\{M_2, Y_2\}$.
For a query $Send(SM_i, \{M_2, Y_2\})$, SM_i computes $V = e(M_2, S_i){=} e(P, P)^{b+H_3(S_j\|x_2)}$ and verifies the validity of $Y_2{=}H_4(ID_i\|ID_j\|V\|M_2)$. If the result is positive, SM_i calculates $K_{ij} {=}H_2(V^{a+H_3(S_i\|x_1)})$, $sk_{ij}{=}H_4(ID_i\|ID_j\|Z\|V\|K_{ij})$, $G_i{=}H_4(ID_i\|ID_j\|Z\|V\|sk_{ij})$ and returns $\{G_i\}$.
For a query $Send(UC_j, \{G_i\})$, UC_j verifies the validity of $G_i{=}H_4(ID_i\|ID_j\|Z\|V\|sk_{ij})$. If it holds, UC_j shares the session key sk_{ij} with SM_i.
For a query $Execute(SM_i, UC_j)$, it successively returns $\{M_1, Y_1\}$, $\{M_2, Y_2\}$ and $\{G_i\}$ by $Send$ query.
For a query $SSReveal(P)$, $\{a, x_1, Z, M_1\}$ is returned if $P{=}SM_i$ or $\{b, x_2, V\}$ is returned if $P{=}UC_j$.
For a query $SKReveal(P)$, if P has formed a session key sk_{ij} and both its partner and it have not been inquired by a $Test$ query, sk_{ij} is returned. Otherwise, *null* is responded.
For a query $Corrupt(P)$, S_i is returned if $P{=}SM_i$ or S_j is returned if $P{=}UC_j$.
For a query $Test(P)$, sk_{ij} is acquired by query $SKReveal(P)$ and a coin d is flipped. if $d = 1$, a real session key sk_{ij} is returned. Otherwise, a random number selected from $\{0,1\}^l$ is returned.

Case 1. $SSReveal(SM_i)$ and $SSReveal(UC_j)$

E is allowed to get smart meter SM_i's secrecy $\{a,\ x_1,\ Z = g^{a+H_3(S_i\|x_1)}, M_1 = (a + H_3(S_i \parallel x_1))(P_{pub} + H_1(ID_j)P)\}$ and instance UC_j's information $\{b, x_2, V = g^{b+H_3(S_j\|x_2)}\}$.

Case 2. $Corrupt(SM_i)$ and $Corrupt(UC_j)$

E is allowed to get SM_i's long-term private key S_i and the long-term private key S_j of utility control UC_j, but not their ephemeral secrets $\{a,\ x_1,\ b,\ x_2\}$.

Case 3. $SSReveal(SM_i)$ and $Corrupt(UC_j)$

E is allowed to get smart meter SM_i's secrecy $\{a,\ x_1,\ Z = g^{a+H_3(S_i\|x_1)}, M_1 = (a + H_3(S_i \parallel x_1))(P_{pub} + H_1(ID_j)P)\}$ and instance UC_j's secrecy $\{S_j,\ V = g^{b+H_3(S_j\|x_2)}\}$.

Case 4. $Corrupt(SM_i)$ and $SSReveal(UC_j)$

E is allowed to get smart meter SM_i's secrecy $\{S_i,\ Z = g^{a+H_3(S_i\|x_1)}, M_1 = (a + H_3(S_i \parallel x_1))(P_{pub} + H_1(ID_j)P)\}$ and instance UC_j's secrecy $\{b, x_2, V = g^{b+H_3(S_j\|x_2)}\}$.

According to the literature [3], CDHP is equal to DCDHP under the major groups with the prime order q of this letter. For the sake of convenient discussion, we adopt CDHP to denote its equivalent variations. Even if in above four cases, E is unable to compute sk_{ij}, since the subkey K_{ij} cannot be obtained without solving CDHP, whose value is $H_2(g^{(a+H_3(S_i\|x_1))(b+H_3(S_j\|x_2))})$. Thus, so long as CDHP holds, GM_4 and GM_3 are indistinguishable and we can conclude following equation.

$$|Pr[Suc_4] - Pr[Suc_3]| \leqslant q_h Adv_{CDHP}(E) \tag{C5}$$

Game GM_5: This game will be aborted if E asks a $H_4(ID_i\|ID_j\|Z\|V\|K_{ij})$ query. Otherwise, GM_5 and GM_4 are indistinguishable. The probability that E is able to calculate the real sk_{ij} is no more than $\frac{q_h^2}{2^l}$. Thus, we can get following conclusion.

$$|Pr[Suc_5] - Pr[Suc_4]| \leqslant \frac{q_h^2}{2^l} \tag{C6}$$

In addition, from E's point of view, there is no advantage for her to distinguish a random key from the real session key if E does not correctly input a $Hash$ query. Therefore, we can deduce $Pr[Suc_5] = \frac{1}{2}$. To combine all above probabilities, the Theorem 1 holds.

Appendix D Performance analysis

Performance analysis is presented in this section. Here we make comparisons of computation and communication costs to show the performance. Related works [4–6] are selected to compare with our proposed scheme.

We firstly elaborate performance of computation cost. Prior to calculating the computation cost, the execution time of various operations used in schemes should be learned. Here we adopt evaluation results performed by Kilinc et al. [7], which

Table D1 The execution time of various operations

Operations	Time(ms)	Meanings
T_m	2.2260	Execution time of point multiplication
T_b	5.8110	Execution time of bilinear pairing mapping
T_{ph}	12.4180	Execution time of point hash mapping
T_e	3.8500	Execution time of modular exponentiation
T_s	0.0046	Execution time of symmetric en/decryption
T_h	0.0023	Execution time of one-way hash function

Table D2 Comparison of computation cost with related schemes

Schemes	Smart meter(ms)	Utility control(ms)	Total cost(ms)
[4]	$3T_m + T_b + 2T_{ph} + T_s + 5T_h$=37.3319	$3T_m + T_b + T_e + 2T_{ph} + T_s + 5T_h$=16.3597	78.5332
[5]	$4T_m + T_e + 5T_h$=12.7655	$3T_m + 2T_b + T_e + 5T_h$=22.1615	34.9040
[6]	$2T_m + T_b + T_e + 3T_h$=14.1199	$2T_m + 2T_b + T_e + 4T_h$=19.9332	34.0531
Ours	$4T_m + T_b + 2T_e + 7T_h$=22.4311	$4T_m + T_b + 2T_e + 7T_h$=22.4311	44.8622

are based on specifications of dual CPU E2200 2.2 GHz, 2.0 GB main memory, and Ubuntu operating system. Time to execute different operations and their meanings are shown in Table D1. Subsequent result of comparison on computation cost is shown in Table D2. Since the time for executing operations of XOR and point addition is negligible, the computation cost of them is not included. As Table D2 shows, the total operating time of our scheme is moderate and reaches 44.8622ms, which is higher than [6] but still lower than [4].

Table D3 Comparison of communication cost with related schemes

Schemes	Number of messages	Total cost(bits)
[4]	4	960
[5]	3	1408
[6]	3	1632
Ours	3	1120

Next, we evaluate the communication cost. Assuming that the speed of message sending is the same, then the communication cost relies on the bit length of total messages. Here we adopt the setup of bit size assumed in [2]. The random number and time-stamp are supposed to 128 bits and 32 bits in size, respectively. The identity and hash output for transmission are supposed to 160 bits in size. In addition, let elements in G_1 be 320 bits in size and let elements in G_2 be 512 bits in size. Subsequently, we take our scheme as an example for illustration. In the proposed scheme, three messages are sent after a complete agreement is executed. The transmitted messages include Mes_1=$\{M_1, Y_1\}$, Mes_2=$\{M_2, Y_2\}$, and Mes_3=$\{G_i\}$. Communication cost of Mes_1 and Mes_2 are both equal to 160+320=480 bits, and Mes_3 requires 160 bits of communication cost. Thus, the total cost in communication is 480+480+160=1120 bits. The comparison on communication cost is presented in Table D3. From Table D3, we are able to draw a conclusion that our scheme requires only 1120 bits and is lower than compared schemes when the number of messages is the same.

Although Mahmood et al.'s scheme [6] outperforms our scheme from the perspective of computation cost, our scheme requires lower communication cost and enhances the security of theirs. Therefore, our scheme is more appropriate for application in practice.

References

1 Canetti R, Krawczyk H. Analysis of key-exchange protocols and their use for building secure channels. In: International Conference on the Theory and Applications of Cryptographic Techniques, Berlin: Springer, 2001. 453-474.

2 Odelu V, Das A K, Wazid M, et al. Provably secure authenticated key agreement scheme for smart grid. IEEE Transactions on Smart Grid, 2016, 9: 1900-1910.

3 Bao F, Deng R H, Zhu H. Variations of Diffie-Hellman problem. In: International Conference on Information and Communications Security, Berlin: Springer, 2003. 301-312.

4 Wang Y. Password protected smart card and memory stick authentication against off-line dictionary attacks. In: IFIP International Information Security Conference, Berlin: Springer, 2012. 489-500.

5 Tsai J L, Lo N W. Secure anonymous key distribution scheme for smart grid. IEEE Transactions on Smart Grid, 2015, 7(2): 906-914.

6 Mahmood K, Li X, Chaudhry S A, et al. Pairing based anonymous and secure key agreement protocol for smart grid edge computing infrastructure. Future Generation Computer Systems, 2018, 88: 491-500.

7 Kilinc H H, Yanik T. A survey of SIP authentication and key agreement schemes. IEEE Communications Surveys & Tutorials, 2013, 16: 1005-1023.