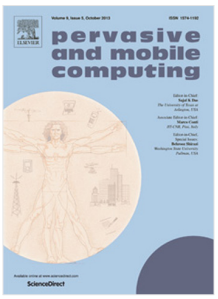
Accepted Manuscript



A secure authentication scheme for Internet of Things

King-Hang Wang, Chien-Ming Chen, Weicheng Fang, Tsu-Yang Wu

PII: S1574-1192(16)30436-9
DOI: https://doi.org/10.1016/j.pmcj.2017.09.004
Reference: PMCJ 889

To appear in: *Pervasive and Mobile Computing*

Received date : 14 December 2016
Revised date : 24 August 2017
Accepted date : 1 September 2017

Please cite this article as: K. Wang, C.-M. Chen, W. Fang, T. Wu, A secure authentication scheme for Internet of Things, *Pervasive and Mobile Computing* (2017), https://doi.org/10.1016/j.pmcj.2017.09.004

A Secure Authentication Scheme for Internet of Things

King-Hang Wang

Hong Kong University of Science and Technology, Hong Kong

Chien-Ming Chen, Weicheng Fang

Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

Tsu-Yang Wu

Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fujian, China

National Demonstration Center for Experimental Electronic Information and Electrical Technology Education, Fujian University of Technology, Fujian, China

Abstract

Security is one of the major issues in Internet of Things (IoT) research. The rapid growth in the number of IoT devices, the heterogeneity and complexity of these objects and their networks have made authentication a challenging task. Other constraints such as limited computational ability and power, and small storage of some embedded devices make implementation of complex cryptographic algorithms difficult. So far there has been no established industrial standard to address this problem.

Recently, Kalra and Sood, and subsequently Chang *et al.* attempted to solve the authentication problem by proposing key agreement schemes for IoT devices. However, the security of their schemes were unproven. In this paper we demonstrate that these schemes are insecure. We extend upon their work to present a scheme that enables embedded devices to communicate securely with a server on an IoT network. We prove the security of this scheme using formal methods and demonstrate this under the intractability of some well-defined hard problems. We also discuss some practical aspects related to the implementation of the scheme.

Keywords: IoT Security, Authentication, Elliptic Curve Cryptography,

Cryptanalysis

A Secure Authentication Scheme for Internet of Things

King-Hang Wang

Hong Kong University of Science and Technology, Hong Kong

Chien-Ming Chen, Weicheng Fang

Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

Tsu-Yang Wu

Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fujian, China

National Demonstration Center for Experimental Electronic Information and Electrical Technology Education, Fujian University of Technology, Fujian, China

1. Introduction

The Internet of Things (IoT) [1] has opened up a new opportunity to connect people with sensors and devices distributed around their physical world. Embedded devices, wearable sensors, RFID enabled devices, and other "things" can interact on the IoT network to enable better connection and data exchange. This has led to new avenues for connecting technologies and businesses in the areas of commerce, health care, transportation and domotics. In recent years this has made the IoT a popular topic of research in both academia [2, 3] and industry. A business report from Deloitte [4] cites a prediction that by 2020 there will be over 1.9 trillion dollars worth of economic value addition from 26 billion devices on the IoT.

The Internet of things has become so seamlessly integrated into everyday life, that often the end users are completely unaware of the presence of these devices around them. This invisibility of IoT devices has made managing their security extremely important yet challenging. In recent times several IoT networks have been taken over to carry out malicious attacks on the internet. Researchers and industry experts (as in the Deloitte report) recognize that tackling these

security issues of IoT is of high priority for their successful development and adoption.

There has been a rapid increase in the number of devices [5] in the IoT world. Providing authentication (for proper access control) for all these devices is very challenging - the objects, schemes and networks on the IoT are heterogeneous and complex. They are often limited by limited computational ability, minimal storage and constrained power. This makes it extremely difficult to implement traditional cryptographic solutions to protect these devices. Some research works have evolved in this area [6, 7, 8, 9, 10, 11] to tackle these challenges, but there is still a lack of an industrial standard to provide an efficient, secure, and easy-to-manage authentication scheme for the IoT. The only IoT standard which reaches close to addressing this issue is the 6LoWPAN [12] which enables compression of IPv6 header and encryption frames with shared pairwise keys. However, the keys are pre-established and how devices should be authenticated have not yet been addressed by the standard [13].

To address the issue, in 2015 Kalra and Sood [14] proposed a mutual authentication and key agreement scheme for IoT devices and cloud servers based on the elliptic curve cryptography algorithm using encrypted cookies. However, Chang *et al.* [15] pointed out a major flaw in this scheme to show that it is not possible to achieve mutual authentication (allowing the server and devices to authenticate each other) and session key agreement (producing a computable session key for encrypting their communication) as claimed. Subsequently Chang *et al.* proposed an improved scheme which attempted to overcome the flaws in the work of Kalra and Sood. In this paper we demonstrate that the approach proposed by Chang *et al.* is still insecure and cannot deliver the security as claimed. An adversary without processing any secret information can impersonate an honest server and establish a connection with an embedded device. In this paper we attempt to explore why these cryptographic schemes are insecure. We show the need to use a formal model to prove the security of a scheme - an approach which had not been used by the mentioned papers.

Under a formal model, an attacker's possible behaviors are formulated as a

set of interactions with devices and the server. Making use of this reduction enables us to prove that breaking a scheme is indeed the same as breaking some well-defined cryptographic assumptions. Thus, in this paper, we not only present an improvement over Kalra and Sood's and Chang *et al.*'s works, but also show that our scheme is secure under a formal modeling technique that is widely accepted in the literature.

The contributions of this paper are summarized as follows:

1. We demonstrate that the scheme proposed by Chang *et al.* is insecure.
2. We propose a secure scheme for the IoT environment and prove its security under the random oracle model.
3. We present a formal proof of our scheme in order to prevent the subsequent scheme cracking loop.

The rest of this paper is organized as follows. In Section 2 we review some preliminary literature in cryptography. In section 3 we provide reviews of both Kalra & Sood's, and Chang et al.'s schemes. In section 4 we demonstrate why Chang et al's scheme is insecure. An amendment is proposed in Section 5 and a discussion on the security is given in Section 6. Some practical aspects of the scheme is provided in Section 7.

2. Preliminaries

2.1. Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a technique that provides a high level of security in public key encryption and digital signatures. A major advantage of ECC over traditional modulo arithmetic based approaches such as RSA or ElGamal is that it requires a shorter key and a lower computational effort[16]. In the light of the fact that RSA factorization is breakable using quantum algorithms [17], ECC has gained a wide scale industry adoption and has become an important industry standard in recent years. The ECC works as defining each coordinate or point on a elliptic curve (for example, $y^2 = x^3 + 117050x^2 + x \mod 2^{221} - 3$ as an additive group element).

In this additive group addition ($+$) and scalar multiplication ($\times$) are defined and can be computed in a very efficient way. Given G is an element on an additive group, we say G is a *generator* over this group if $O, G, 2G, 3G, 4G, \ldots, (n-1)G$ are all distinct and form all possible elements in this group. In this case, $nG = O$ where O is the identity element in this group. The size of this group, or known as the order of the group, is n.

However, with a pair of given elements P and kP (which means the scalar multiplication of P by an integer k), it is intractable to compute the value k. We refer to this as the *Elliptic Curve Discrete Log Problem* (ECDLP). Furthermore, it is hard to computer abP by given aP, bP and P only. We refer this hard problem as the *Elliptic Curve Computation Diffie-Hellman Problem* (ECCDH, or simply CDH). Please refer to the appendix for more detail descriptions of the ECDLP and ECCDH.

2.2. Adversary Model

The adversary model is described as follows:

A polynomial time adversary $\mathcal{A}$ who has full control over the network traffic desires to break the security of the scheme. $\mathcal{A}$ may initiate a scheme with any embedded device or the server. $\mathcal{A}$ can eavesdrop on the data sent between an embedded device and the server, inject messages between them, or replay the transmitted messages. The goal of $\mathcal{A}$ is to achieve one of the following:

1. Have the embedded device or the server falsely accept an authentication scheme when they are not communicating with a legitimate entity,
2. Compute the session key generated between an embedded device and the server after a successful run of the authentication scheme, or
3. Compute the long-term secret key of the server or an embedded device.

3. Reviewing KS-scheme and CWS-scheme

In this section we will review the original scheme proposed by Kalra and Sood, which we refer to as the *KS-scheme*. We also review the amended version presented by Chang *et al.*, which we refer to as the *CWS-scheme*.

In Table 1 we define the notations used in the subsequent discussions.

Symbol	Explanation
D_i	An embedded device registered in the system.
$\mathcal{D}$	The set of devices, $\mathcal{D} = \{D_1, D_2, \cdots, D_i\}$
S	The server.
$\mathcal{A}$	An attacker.
ID_i	The ID number of the device D_i.
$\mathcal{PW}_i$	The password for the device D_i.
H	A cryptographic hash function with an output of l_H bits. $H : \{1\}^* \rightarrow \{1\}^{l_H}$.
h	A cryptographic hash function with an output of l_h bits. $h : \{1\}^* \rightarrow \{1\}^{l_h}$.
$\mathcal{G}$	An additive group implemented by an elliptic curve.
G	A generator of the group $\mathcal{G}$. A public parameter.
EXP_Time	The expiry time of a particular device.
X	Server's secret key.
SK	A session key outputed at the end of a scheme.

Table 1: Notation Table- This contains the notations used in reviewing the Kalra and Sood, and Chang el al. schemes.

3.1. KS-Scheme

In this scheme, there is a single trusted server S and several embedded devices $\mathcal{D}$. The server keeps the secret X and it is not shared with any other devices. It picks some system parameters: a cryptographic hash function $H(.)$, a elliptic curve E, a generator G on E and publishes them as public parameters. An embedded device $D_i \in \mathcal{D}$ will need to perform a one-time registration with the server. It selects its own ID ID_i and sends it to the server. The server then generates a unique password PW_i, a random number R_i and computes the values of $\{CK, CK', T_i, A_i, A_i'\}$ shown as below.

$$CK = H(R_i||X||EXP\_Time||ID_i) \tag{1}$$

$$CK' = CK \times G \tag{2}$$

$$T_i = R_i \oplus H(X) \tag{3}$$

$$A_i = H(R_i \oplus H(X) \oplus \mathcal{PW}_i \oplus CK') \tag{4}$$

$$A_i' = A_i \times G \tag{5}$$

The values of $\{A_i', T_i, ID_i, EXP\_Time\}$ are securely stored in the database of the server. The value of CK' is transmitted to the embedded device D_i and D_i

securely stores the value. Besides, both the server and D_i need to store the public parameter G.

When D_i attempts to login to the server, it generates a random number N_1 and computes the values of P_1, P_2 and sends them together with the ID ID_i to the server as below.

$$P_1 = N_1 \times G \tag{6}$$

$$P_2 = H(N_1 \times CK') \tag{7}$$

While receiving P_1 and P_2, the server retrieves $\{A'_i, T_i, ID_i, EXP\_Time\}$ from its database and calculates $CK = H(R_i||X||EXP\_Time||ID_i)$ where $R_i = T_i \oplus H(X)$. It then calculates $P'_2 = H(P_1 \times CK)$. If the value of P'_2 is equals to P_2, the server replies to the device ID_i with T_i, P_3, P_4 using the equation below:

$$P_3 = N_2 \times G \tag{8}$$

$$P_4 = N_2 \times A'_i \tag{9}$$

The embedded device then validates the correctness of P_4 by comparing the value of P_4 and P'_4 where $P'_4 = P_3 \times A_i$ and $A_i = H(T_i \oplus PW_i \oplus CK')$. Finally it calculates V_i to demonstrate its knowledge of CK' to the server.

$$V_i = H(CK' \times N_1||P_4) \tag{10}$$

The server also verifies V_i using its knowledge of CK, P_1 and P_4 and the equation:

$$V_i \stackrel{?}{=} H(CK \times P_1||P_4) \tag{11}$$

Now both the server and D_i generate a session key using $SK = H(X||ID_i||N_1||N_2)$. The KS-scheme can be summarized as in Fig 1.

Unfortunately, Chang *et al.* pointed out that the session keys are not computable in *KS-scheme*. This is because devices have no knowledge about the value of X. Besides, the device D_i cannot compute P'_4 because it has no knowledge about the value of PW_i.

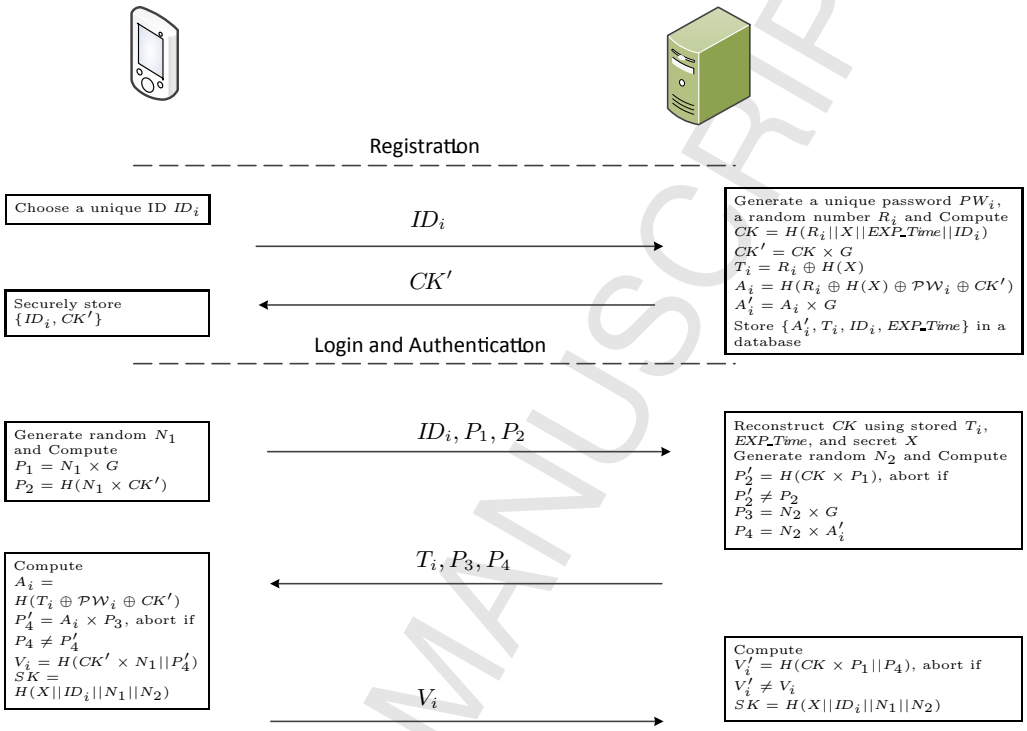


Figure 1: Illustration of KS-scheme.

3.2. CWS-Scheme

The CWS-Scheme is presented in Fig 2. The CWS-scheme tries to address the problems of KW-scheme while minimizing the change required. The changes are highlighted in red in Fig. 2. The first change is that a password (or a passphrase-key) $\mathcal{PW}_i$ is chosen by the server and securely transmitted the hash of the password $H(\mathcal{PW}_i)$ to the embedded device D_i during the registration step for further computations.

The second change is that the computation of the value of A_i at different stages is performed using the value of $H(\mathcal{PW}_i)$ which was previously stored. To differentiate between the symbols, we denote $\hat{A}_i$ as the A_i which is used in the CWS-scheme. That is:

$$\hat{A}_i = H(R_i \oplus H(X) \oplus H(\mathcal{PW}_i) \oplus CK'). \tag{12}$$

At the end of the scheme, a session key SK is computed.

Note that the parameters $H(.), E, G$ remains public as the KS-scheme. With the applied changes, the authors claimed that their scheme allows the parties to mutually authenticate each other and establish a secure session key. Unfortunately, we found that the application of only these two changes is not sufficient to fix the KS-Scheme.

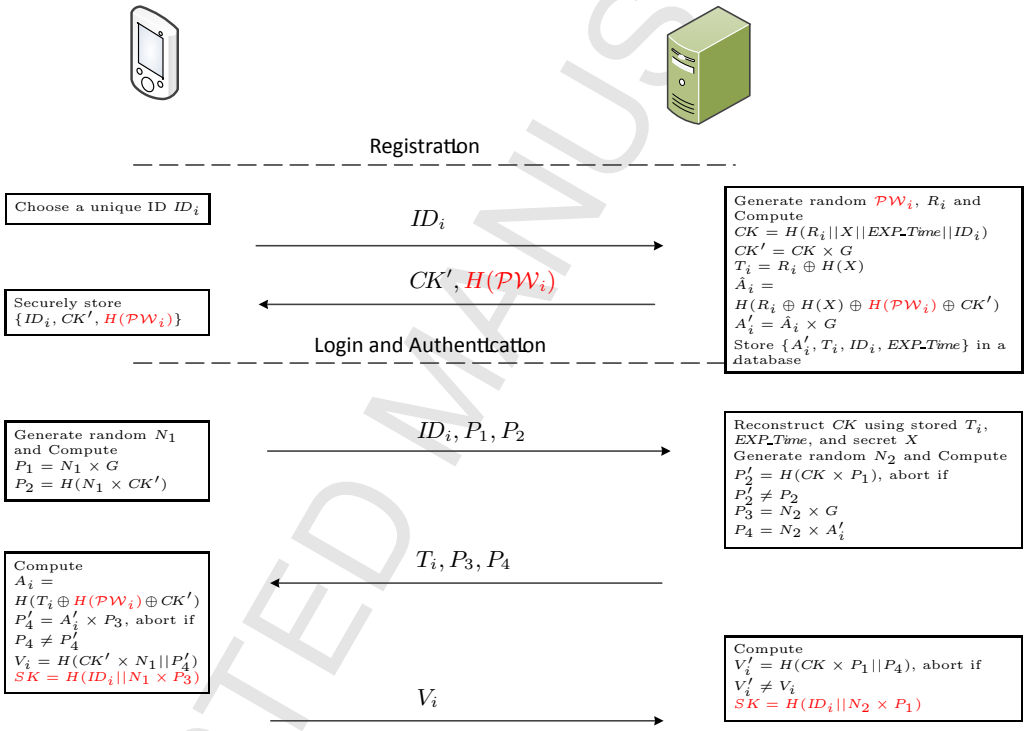


Figure 2: Illustration of CWS-scheme. The improvement over KS-scheme are highlighted in red.

4. Analysis of the CWS-scheme

In this section we identify a security attack on the CWS-scheme. We allow the adversary to have control over the communication channel; thus, that the adversary can eavesdrop on, inject, delete, or modify any message. The goal of the adversary is to 1) make either one of the parties (either the server or the

client) to mistakenly believe they are connecting with an honest partner while in reality it is the attacker; or 2) achieve a significant probability to accurately compute the session key agreed between the two honest parties.

Under the CWS-scheme an adversary may *impersonate* an honest server and establish a connection with a embedded device D_i (i.e. accomplish goal 1). Prior to the attack, the adversary would need to obtain the value of T_i, P_3 and P_4, which can be obtained by logging a previous scheme execution between D_i and the service. When the device D_i tries to login to the server the next time, the adversary can impersonate the server and interact with the device D_i. On receiving the messages ID_i, P_1, and P_2 in the first step of the scheme, the adversary simply responds with the previously logged values of T_i, P_3 and P_4 to the devices D_i. Since $P_4 = A_i \times P_3$, they will pass the device D_i's validation step. Besides, since there are no more validation messages required from the server, D_i will therefore always accept this scheme and mistakenly identify the attacker as the server and assume that the established connection is secure.

Despite the fact that the attacker does not hold the session key and is unable to further communicate with the device, the impact of this attack can be very severe on certain applications. Consider that a server is providing software security patching service to a mobile device. After established the connection with a fake server, the mobile device will mistakenly believe there is no software needs to update. This also applies to natural disaster alarm service where "no message" from a trusted server could be a much serious problem than denial-of-services.

Even if the device D_i is very cautious and compares the value of P_3, P_4 against its login records to filter any of these simple replay attacks, there are still other vulnerabilities which the attacker can exploit. In such cases, the adversary only needs to randomly generate a number α and multiply α with P_3 and P_4 such that $\hat{P}_3 = \alpha \times P_3$, $\hat{P}_4 = \alpha \times P_4$, and then reply to the devices D_i with the messages T_i, $\hat{P}_3$, and $\hat{P}_4$. Again, $\hat{P}_4 = \alpha \times P_4 = \alpha \times A_i \times P_3 = A_i \times \hat{P}_3$ which is indistinguishable from the response of an authentic server.

To further communicate with the device and create a larger damage, the

adversary needs to calculate the session key (i.e., accomplish goal 2). To derive the session key SK the adversary needs $ID_i, N_2 \times P_1$ where ID_i and P_1 are sent over the air. Unfortunately N_2 cannot be computed from P_3 and G although $P_3 = N_2 \times G$. This is a computation hard problem – ECDLP.

Instead the adversary can utilize the concept of the small group confinement attack [18]. Small group confinement attacks have been developed 20 years ago where it is used to attack Diffie-Hellman based system. Similar techniques can be used here to compute the session key SK. Assume the group order of the elliptic curve is n where n has some prime factors $q_1, q_2, q_3, \cdots$ so that q_1 is a relatively small number and looping on q_1 is computationally feasible. Given G is a generator and the group order of the elliptic curve is n, we have:

$$n \times G = O,$$

where O is the identity element on the curve such that $n \times O = O$. If G is multiplied by n/q_1, the order of this number would becomes q_1, since

$$q_1 \times (\frac{n}{q_1} \times G) = O.$$

Thus, any number x multiplies that number can be viewed as $(x \mod q_1)$ multiplies that number, that is,

$$\begin{aligned} x \times (\frac{n}{q_1} \times G) &= (k(q_1) + x \mod q_1) \times (\frac{n}{q_1} \times G) \\ &= k(q_1) \times (\frac{n}{q_1} \times G) + (x \mod q_1) \times (\frac{n}{q_1} \times G) \\ &= (x \mod q_1) \times (\frac{n}{q_1} \times G) \end{aligned}$$

Following this concept, $\mathcal{A}$ receiving P_3, P_4 in a protocol forges $\hat{P}_3, \hat{P}_4$ by

$$\begin{aligned} \hat{P}_3 &= \frac{n}{q_1} \times P_3 = \frac{n}{q_1} \times N_2 \times G, \\ \hat{P}_4 &= \frac{n}{q_1} \times P_4 = \frac{n}{q_1} \times N_2 \times A_i'. \end{aligned}$$

$\hat{P}_3, \hat{P}_4$ will be authenticated while the SK computed is now become enumerable. Since the only unknown component of SK is $N_1 \times \hat{P}_3$ and $\hat{P}_3$ is in the subgroup

with order q_1, $N_1 \times \hat{P}_3$ has only q_1 possibility and can be easily enumerated in a loop by verifying any encrypted message sent from the sender.

Similar to the CWS-scheme, the KS-scheme is also vulnerable to the above attacks with the exactly same method we break the CWS-scheme, in additional to the problems identified by Chang *et al.*.

As we can see both schemes are insecure. The impact of this vulnerability allows an adversary to make any victim embedded device falsely believe that they have connected to a server. Sensitive data can be sent to the malicious server and the malicious server may issue some improper commands to the embedded devices.

Apart from the attack mentioned above, the schemes may be vulnerable to other attacks or the variants of the above-mentioned attacks. Therefore, a break-and-fix-and-break-and-fix cycle will probably continue forever if the patching scheme is not proven secure. In this paper we would like to propose a new scheme that is provable secure.

5. The proposed scheme

In this section we present the amended version of the scheme to make it more secure against an active attack. We maintain the original philosophy of the KS-scheme in order to adapt it to the IoT environment, while at the same time adapt it to the standard security model in an open network environment. There are few design issues we would like to point out before describing our scheme.

1. **Password**. We note that the password/passphrase PW_i (in CWS-scheme) has no real security function. This secret is stored together with the value CK' at the client side. When the PW_i is stolen by an attacker it is reasonable to assume that the CK' will also be stolen. Even if PW_i is a human-memorable password and is not stored on the device, an attacker who cracks the device and obtains the value CK' will be able to login to the server without the need of PW_i.

2. **Length of variables.** It is not explicit in either schemes what the length of variables and the output of the hash function should be, and how they may impact the security of the schemes. Consider an extreme example, in which the server secret key X and the random number R_i are only 20-bits random variables. An attacker may launch an off-line guessing attack to break the entire system. Thus, the length of each variable, as we will see later, plays an important role in providing the security of our formal model.

3. **Cryptographic Hardness.** In the original paper by Kalra and Sood, they make the assumption about the cryptographic hardness of the Elliptic Curve Discrete Log Problem (ECDLP). The assumption states that a polynomial would not be able to compute the value a, given the points aG, G on a carefully chosen elliptic curve $\mathcal{G}$. Yet, we shall see that in both the CWS and our scheme, a stronger assumption, the Elliptic Curve Computational Diffie-Hellman[16] (ECCDH) is required.

5.1. Description of the Scheme

Here we adopt a set of notations and make a set of assumptions. They are:

A cryptographic hash function $H(.)$ randomly maps an arbitrary string to a l_H-bit long string. A random function $h(.)$ is assumed to randomly map an arbitrary string to a l_h-bit long string. We consider an elliptic curve to be an additive group $\mathcal{G}$ and G is an element of this group. The order of G is at least 2^{l_h}.

Our scheme improves upon the CWS-scheme by removing the password and calculating the values of P_2, P_4, V_i and SK in a different way. Fig 3 illustrates the details of our scheme. The server has a long term secret key X with l_h-bit long. An embedded device D_i first registers its ID_i with the server. The server then computes the following values:

$$CK = h(R_i||X||EXP\_Time||ID_i) \tag{13}$$

$$CK' = CK \times G \tag{14}$$

$$T_i = R_i \oplus H(X) \tag{15}$$

$$A_i = h(R_i \oplus H(X)||CK') \tag{16}$$

$$A_i' = A_i \times G. \tag{17}$$

The server then stores $\{A_i', T_i, ID_i, EXP\_Time\}$ in its database. The value of CK' is delivered to the embedded device over a private channel and D_i securely store the values of ID_i and CK'.

When the embedded device desires to be authenticated by the server, it selects a l_h-bit long random number N_1 and computes the values of P_1, P_2 as follows:

$$P_1 = N_1 \times G \tag{18}$$

$$P_2 = H(P_1||N_1 \times CK'). \tag{19}$$

$\{P_1, P_2\}$ are sent to server for authentication. The server retrieves the stored record associated with ID_i from its database. It validates the value of P_2 using its knowledge of X, T_i, EXP_Time and the given the value of P_1. Then it chooses a l_h-bit long random number N_2 and computes the values of P_3, P_4 using the formulas:

$$P_3 = N_2 \times G \tag{20}$$

$$P_4 = H(P_2'||N_2 \times A_i'). \tag{21}$$

$\{T_i, P_3, P_4\}$ are transmitted to the device for authentication. The device then validates the value of P_4 by first reconstructing A_i from T_i and CK'. If the value of P_4 is legitimate, then the value of $H(P_2||A_i \times P_3)$ would be the same

as $H(P_2||A_i \times N_2 \times G)$, which is equal to $H(P_2||A' \times N_2)$. Finally the device computes the values of V_i and SK as follows:

$$V_i = H(P_4'||N_1 \times P_3) \tag{22}$$

$$SK = H(P_3||N_1 \times P_3). \tag{23}$$

V_i is sent to the server for authentication and SK is the agreed value of the secret key during this session. The server would be able to compute the same values of V_i and SK by considering the identity $N_1 \times P_3 = N_2 \times P_1 = N_1 \times N_2 \times G$.

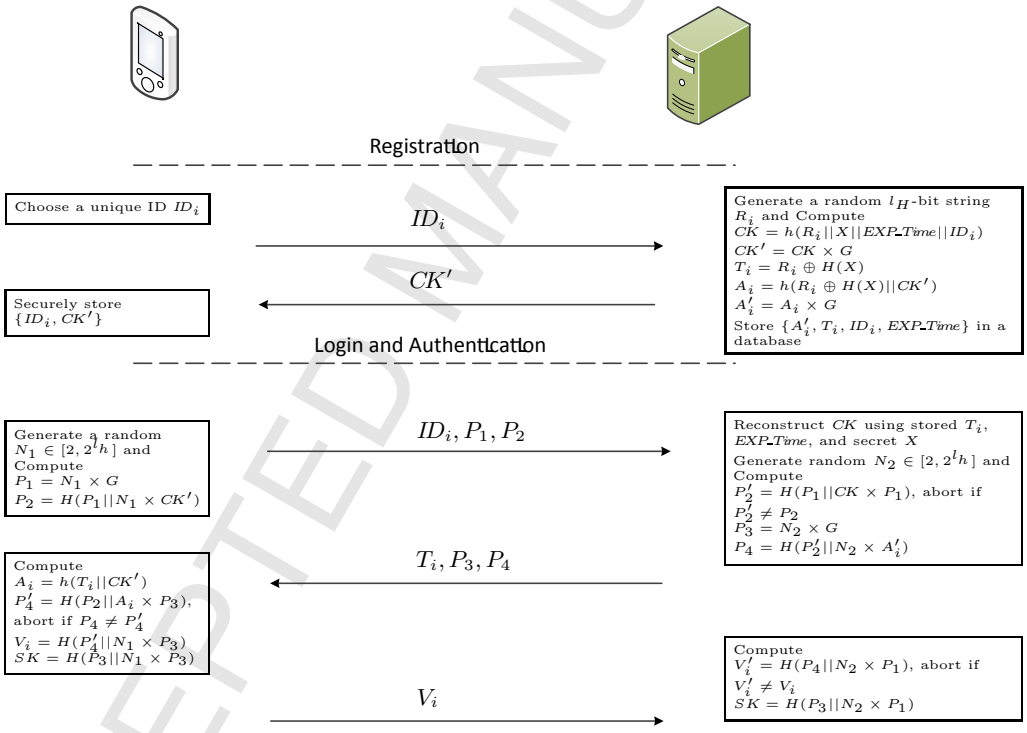


Figure 3: Illustration of our proposed amendment.

6. Security Analysis and Formal Proof

In order to break the scheme cracking loop in the literature, there is a need to establish that the proposed scheme is indeed as secure as it claims to be.

To demonstrate its soundness, we present both a heuristic security analysis and a formal proof of the security of the scheme. We adopt the random oracle model [19] and follow the semantics which are commonly used in the literature of provable security [20, 21, 22]. In the random oracle model we consider a one-way hash function (or simply hash function) as a random function which the adversary cannot invert or crack. We model the hash function as an *oracle* for the adversary to query. The security proof rests on the assumption that all queries that help the adversary to break the scheme are recorded. By tracing the queries we can solve a hard problem. However, the hard problem cannot be solved, and hence by contradiction the adversary must not be able to break to scheme.

6.1. Heuristic Security Analysis

In our scheme the major design changes we implement are as follows: a) We remove the unused password. b) We ensure the freshness of variables to prevent a message replay attack. We do this by ensuring that each message in the scheme serves a new challenge to the partner and the partner is required to respond to each fresh challenge.

We argue that the scheme is secure against the following attacks:

Replay Attack. First of all we observe that the scheme no longer suffers from the problem of replay attack demonstrated earlier. If the adversary tries to impersonate the server again, he will need to present P_4 which is composed of a secret A_i' and a fresh challenge P_2 . Since this fresh challenge is unavailable to the adversary, the attack will fail. Similarly, if the adversary wishes to impersonate a device by replaying the messages P_1 and P_2 from the previous round of the scheme, it would be required to produce V_i which contains the knowledge of N_1 which is a discrete log of P_1 and a fresh challenge P_3.

Stolen Session Key Attack. Suppose an adversary is able to recover a set of session keys SK; under our security scheme he will be unable to a) recover the message CK' required to login to the server or b) to impersonate a server to trick a device in an later on session. This is due to the fact that SK is produced

by a secure hash function from a temporary Diffie-Hellman secrets. Therefore recovering session keys does not give any advantage to adversary to the recover the long term secret key.

Perfect forward secrecy. Let us assume that either the long term secret CK' or the server secret is exposed to the adversary. If this happens, it would not provide any advantage to the adversary to recover the session keys established previously. The reason is that all previously produced session keys SK were generated with the short term Diffie-Hellman secrets. If the secrets were deleted securely (which must be taken care of in the software implementation), there is no way for the attacker to deduce the shared secret key from the logged communication needed to solve the CDH.

The security of the system to the above attacks mechanisms are proven formally in the following section.

6.2. Formal Proof

Here we formulate the activities and goals of the adversary as a *game*. We say that the scheme is secure if and only if the probability of the adversary $\mathcal{A}$ *winning* the game is negligible.

Formally, we define the game as below:

1. The game has the following participants: an attacker $\mathcal{A}$, a scheme $\mathcal{P}$, a server S, a set of devices $\mathcal{D} = \{D_1, D_2, \cdots, D_i\}$, and some system parameters l_H, l_h. Every device is assumed to have been already registered with the server at the beginning.
2. $\mathcal{A}$ initiates the scheme $\mathcal{P}$ with the $\mathcal{D} \bigcup S$ by invoking a query expressed as **Execute**(D_i, S). This query instructs a device D_i to log on to the server S. The participants shall respond to the query and output the reply message to a transcript shared with $\mathcal{A}$.
3. $\mathcal{A}$ may issue a query **send**(Π, M) to the game which is modeling the adversary $\mathcal{A}$, by sending a message M to a scheme participant Π (can be either D_i or S). The participant will respond to the $\mathcal{A}$ according to the specifications of the scheme.

4. $\mathcal{A}$ may issue a query **corrupt**(D_i) to gain access to the secret which is securely stored by CK' stored in the device D_i. This models an insider attack or a stolen-device attack.
5. $\mathcal{A}$ may issue a query **reveal**(D_i) to obtain a session key SK established during the scheme.
6. After the previous round of the scheme, assuming that a fresh participant[1] Π does not terminate in the middle, he will generate a session key. $\mathcal{A}$ may issue a query **test**(Π) to ask Π to return the calculated key. The game will flip a fair coin $\mathcal{C}$. If the value of $\mathcal{C} = 0$ (head), Π will return the session key to the adversary; if $\mathcal{C} = 1$ (tail), Π will return a random string with the same distribution as the session key to $\mathcal{A}$.
7. The goal of $\mathcal{A}$ is to correctly guess the value of $\mathcal{C}$. $\mathcal{A}$ will output a bit τ at the end of the game as the guess of the value $\mathcal{C}$. In the game, the $\mathcal{A}$ wins if $\tau = \mathcal{C}$. If the value does not match $\mathcal{C}$, $\mathcal{A}$ will lose the game.

The interaction specified above highlights the capabilities of an attacker described in Section 2. The interaction even accounts for the case where the adversary may tamper with the participating devices by issuing a **corrupt** query, or hack an established session by issuing a **reveal** query.

Definition 1. ***Security***. *Let $Win(\mathcal{A}, \mathcal{P})$ be the event in which an adversary $\mathcal{A}$ wins the above game for the scheme $\mathcal{P}$. A scheme $\mathcal{P}$ is considered to be secure if for any polynomial time, the probability of an $\mathcal{A}$ winning given by $\Pr(Win(\mathcal{A}, \mathcal{P}))$ is bounded by $1/2 + \epsilon$ where the value of ϵ is a negligible.*

Theorem 1. *Let $\mathcal{P}$ be the scheme described in above section. If the CDH assumption holds, then we can state that the scheme is secure. This can be written as:*
$\Pr(Win(\mathcal{A}, \mathcal{P})) \leq \frac{{q_H}^2 + (q_s + q_e)^2}{2^{l_H}} + \frac{q_h + q_e + q_s + 3q_H}{2^{l_h}} + 1/2 + CDH$
*where q_s, q_e, q_H, q_h denote the numbers of **Send** queries, **Execute** queries, and hash queries to H and h respectively and CDH is the probability of solving the CDH instance.*

The proof of Theorem 1 is available in the appendix.

[1] A fresh participant is one who is neither the current participant nor the partner who has been *corrupted* or *revealed* by $\mathcal{A}$.

7. Practical Issue

In this section we discuss some practical issues related to this scheme. To make the discussion more concrete, we assume the following parameters are instantiated.

Selection of Elliptic Curve: We select the Curve m-221 [23] for which ECDLP and ECDH are believed to hold.

$$y^2 = x^3 + 117050x^2 + x$$
$$\text{modulo } p = 2^{221} - 3$$

Here l_h, the bit length of the order of G is 221 bits and l_H, the length of hash $H(.)$, is 128 bits. Using these selected parameters. the theorem 1 shows that the scheme is secure. It is because if the contrary that the scheme can be broken by an adversary with non-negligible probability, it only happens when the adversary can also win the game described in the theorem with non-negligible probability. However the first two terms of the bound have a denominator of 2^{221} and 2^{128} respectively and their nominators are bounded by the square of number of queries. By not considering breaking CDH, with an addition 0.01 winning probability, the adversary needs to perform at least $2^{60.7}$ times of hash functions or scheme communications which is impractical.

The computational effort required by the device is ideal for low powered devices. An Elliptic Curve Diffie-Hellman operation takes only 1.76 seconds and consumes 9.48 mJ, and a SHA-1 operation takes only 0.011 seconds and consumes 57 uJ on a Tmote Sky (8 MHz) node, one of the most common wireless sensor nodes [24, 25]. During the execution of the scheme, it only need to perform 4 hashes and 5 multiplication operations over the curve, which can be completed within 10 seconds on this device[2].

In terms of storage requirements, the device only needs to keep the long term secret CK' and a public parameter G which is are points on the curve.

[2] $1.76 \times 5 + 0.011 \times 4 < 10$

Each of these is only 442 bits long. For security purposes, it does not need to store additional information like the IP address or the public key of the server. During the execution of the scheme, it needs to only temporarily store the values of $P_1, P_2, P_3, P_4, N_1, A_i, V_i, SK$ - in total they amount to only 3.45 Kbits.

This scheme is user friendly as it only requires a one-time setup. In certain application environments, for example the Amazon Dash Button[3], the setup can be done before it is sold to customers. This scheme also allows a re-registration through a simple rerun of the registration scheme and replacement of the value of CK'. Similarly if a device needs to be associated with different servers, it can store multiple tuples for the servers and values of CK' in its database. When a server wishes to blacklist a misbehaving user/device, access revocation can be done by simply rejecting the ID and/or resetting the CK'.

8. Conclusion

In this paper we identify a significant security flaw in the CWS-scheme and present a provable fix to the method. We demonstrate that this fix is robust against different types of attacks, and prove this using a formal method. We believe that the use of formal methods to set up the proof establishes a proper paradigm for the design of a security protocol in the IoT environment.

Acknowledge

This work was supported in part by the Project NSFC (National Natural Science Foundation of China) under Grant number 61402135 and in part by the Shenzhen Foundational Research Project under Grant number JCYJ20150513151706574.

References

[1] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelfflé, Vision and challenges for realising the internet of things, Cluster of European Research Projects on the Internet of Things, European Commision.

[3] Amazon Dash Button, `https://www.amazon.com/oc/dash-button`

[2] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, omputer networks 54 (15) (2010) 2787–2805.

[3] A. Whitmore, A. Agarwal, L. Da Xu, The internet of things—a survey of topics and trends, Information Systems Frontiers 17 (2) (2015) 261–274. doi:10.1007/s10796-014-9489-2. URL http://dx.doi.org/10.1007/s10796-014-9489-2

[4] E. Openshaw, C. Wigginton, J. Hagel, J. S. Brown, M. Wooll, P. Banerjee, The Internet of Things ecosystem: unlocking the business value of connected devices, Technical report, Deloitte.

[5] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, S. Shieh, IoT security: Ongoing challenges and research opportunities, in: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, 2014, pp. 230–234. doi:10.1109/SOCA.2014.58.

[6] D. He, S. Zeadally, An analysis of RFID authentication schemes for internet of things in healthcare environment using elliptic curve cryptography, IEEE internet of things journal 2 (1) (2015) 72–83.

[7] K. T. Nguyen, M. Laurent, N. Oualha, Survey on secure communication protocols for the Internet of Things, Ad Hoc Networks 32 (17–31).

[8] R. Roman, C. Alcaraz, J. Lopez, N. Sklavos, Key management systems for sensor networks in the context of the Internet of Things, Computers & Electrical Engineering 37 (2) (2011) 147—159.

[9] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, X. Wang, TRM-IoT: A trust management model based on fuzzy reputation for internet of things, Comput. Sci. Inf. Syst. 8 (4) (2011) 1207–1228.

[10] J. Liu, Y. Xiao, C. P. Chen, Authentication and access control in the internet of things., in: ICDCS Workshops, 2012, pp. 588–592.

[11] W. Shang, Q. Ding, A. Marianantoni, J. Burke, L. Zhang, Securing building management systems using named data networking, IEEE Network 28 (3) (2014) 50–56.

[12] N. Kushalnagar, G. Montenegro, C. Schumacher, IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals, Tech. rep. (2007).

[13] K.-F. Krentz, H. Rafiee, C. Meinel, 6lowpan security: Adding compromise resilience to the 802.15.4 security sublayer, in: Proceedings of the International Workshop on Adaptive Security, ASPI '13, ACM, New York, NY, USA, 2013, pp. 1:1–1:10.

[14] S. Kalra, S. K. Sood, Secure authentication scheme for IoT and cloud servers, Pervasive and Mobile Computing 24 (2015) 210 – 223, special Issue on Secure Ubiquitous Computing.

[15] C.-C. Chang, H.-L. Wu, C.-Y. Sun, Notes on "secure authentication scheme forIoT and cloud servers", Pervasive and Mobile Computing (2016) –.

[16] D. Hankerson, A. J. Menezes, S. Vanstone, Guide to elliptic curve cryptography, Springer Science & Business Media, 2006.

[17] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM review 41 (2) (1999) 303–332.

[18] D. P. Jablon, Strong password-only authenticated key exchange, SIGCOMM Comput. Commun. Rev. 26 (5) (1996) 5–26.

[19] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: Proceedings of the 1st ACM conference on Computer and communications security, ACM, 1993, pp. 62–73.

[20] C.-M. Chen, K.-H. Wang, T.-Y. Wu, J.-S. Pan, H.-M. Sun, A scalable transitive human-verifiable authentication protocol for mobile devices, IEEE Transactions on Information Forensics and Security 8 (8) (2013) 1318–1330.

[21] E. Bresson, O. Chevassut, D. Pointcheval, Provably authenticated group diffie-hellman key exchange – the dynamic case, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2001, pp. 290–309.

[22] D. He, S. Zeadally, N. Kumar, J. H. Lee, Anonymous authentication for wireless body area networks with provable security, IEEE Systems Journal PP (99) (2016) 1–12. doi:10.1109/JSYST.2016.2544805.

[23] D. F. Aranha, P. S. L. M. Barreto, G. C. C. F. Pereira, J. E. Ricardini, A note on high-security general-purpose elliptic curves, Cryptology ePrint Archive, Report 2013/647, http://eprint.iacr.org/2013/647 (2013).

[24] A. Liu, P. Ning, Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks, in: Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on, IEEE, 2008, pp. 245–256.

[25] F. Kausar, S. Hussain, J. H. Park, A. Masood, Secure Group Communication with Self-healing and Rekeying in Wireless Sensor Networks, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 737–748.

Appendix A. Proof of Theorem 1

Before introducing the proof, it is essential to state the cryptographic hardnesses:

Definition 2. ***ECDLP Assumption.*** *Assume that there is an elliptic curve such that the points on the curve form an additive group $\mathcal{G}$. G is a generator of this group and kG, represents the summation of G for k times, where kG is an arbitrary point on the curve and k is an arbitary integer. The ECDLP assumption holds if, given an instance ($\mathcal{G}$, G, kG), it is infeasible to compute k.*

Similar to the computational Diffie-Hellman assumption in modulo arithmetic, regarding the hardness of computing $g^{xy} \mod n$ given $g^x \mod n$, $g^y \mod n$, there is a similar hardness assumption about elliptic curves that given

two points aG and bG it is intractable to compute abG. We refer this as the Elliptic Curve Computational Diffie-Hellman Problem (ECCDH).

Definition 3. ***ECCDH Assumption.*** *Let $\mathcal{G}$ be an additive group and G be a generator. The computational Diffie-Hellman assumption holds, if given an instance ($\mathcal{G}$, G, aG, bG) it is infeasible to compute abG without knowing a and b.*

Proof. This proof consists of a sequence of games denoted as $G_j, j \in [0, 3]$ such that these games can make use of an adversary $\mathcal{A}$ with an overwhelming winning probability attempting to solve a CDH instance. However since the CDH assumption holds, by contradiction there cannot exist an adversary $\mathcal{A}$ who has an overwhelming probability of winning. We define Win_j as an event where $\mathcal{A}$ wins the game G_j. A game simulator is constructed as follows to output the solution of a CDH instance $(\mathcal{G}, G, aG, bG)$, i.e , it attempts to compute the value of abG.

Game G_0. This game faithfully follows the scheme specification. All instances of D_i are modeled as the real execution of a random oracle. By definition, we have

$$\Pr(Win_0) = \Pr(Win(\mathcal{A}, \mathcal{P})) \tag{A.1}$$

Game G_1. This game is identical to G_0 except that we simulate the hash function H as follows:

The first step is to prepare a hash list for the hash function H. Whenever a hash function H is invoked by the $\mathcal{A}$ query or by the scheme itself, the game simulator will check if the input has been queried before. If it has not been previously queried, a l_H-bit random string will be generated and both the input and the random string will be recorded in the hash list. In the case that the input has been queried before, it will already be present in the hash list. So the game simulator will simply retrieve and return the stored value. If there is a collision in the hash function, that is, for different inputs it produces the same output, the simulator will terminate and declare that the $\mathcal{A}$ has won. From the perspective of $\mathcal{A}$ this game is no different from the G_0 game till a collision

occurs. Hence

$$|\Pr(Win_1) - \Pr(Win_0)| \leq \frac{{q_H}^2 + (q_s + q_e)^2}{2^{l_H}} \tag{A.2}$$

Game G_2. In this game, we intend to remove the role of the secret X in the server. The game G_2 is identical to the game G_1 except for two differences. The first difference is that T_i is now generated as a random l_H-bit string and R_i is calculated by $T_i \oplus H(X)$. Therefore we can write $A_i = H(T_i||CK')$. This change has no impact on the scheme and is indistinguishable from the perspective of the adversary $\mathcal{A}$. The second difference is that CK is no longer calculated using the hash but is generated as a random l_h-bit string for each individual user. The game is indistinguishable from G_1 until the value of $h(T_i \oplus H(X)||X||EXP\_Time||ID_i)$ is queried by the adversary $\mathcal{A}$. Since the only variable related to X in the scheme is CK', the adversary $\mathcal{A}$ can only guess the value CK' to issue the query in order to distinguish the two games. Hence:

$$|\Pr(Win_2) - \Pr(Win_1)| \leq \frac{q_h}{2^{l_h}}. \tag{A.3}$$

After this game, the server secret key X is unrelated to the scheme and thus the secret key of each device D_i is independent of that of the secret key of other devices.

Game G_3. In this game, the $\mathcal{A}$ must issue a **Test** query to one out of the q_e execution instances. Game G_3 changes an instance of one of the uncorrupted devices to solve a CDH instance. The simulator sets P_1 as $N_1 \times aG$, P_3 as $N_2 \times bG$; and P_2, P_4 as random l_H-bit strings. We argue that this device is uncorrupted and therefore $\mathcal{A}$ does not have the knowledge of CK' or A_i. Unless the adversary $\mathcal{A}$ has ever queried $H(P_1||N_1 \times b \times CK')$ or $h(T_i||CK')$ he cannot differentiate between these changes. Next we consider the probability of this happening under the following scenarios: 1) $\mathcal{A}$ changes the value of P_1 but not of P_3 and P_4 in the communication; 2) $\mathcal{A}$ changes the values P_3, P_4 in the communication; 3) $\mathcal{A}$ does not change any P_i.

1. **Case 1** If $\mathcal{A}$ decides to send a different P_1 to server. In this case a valid P_2 and V_i needs to be produced for authentication. Either the participant

or $\mathcal{A}$ must have queried the hash function H previously to have produced P_2 and V_i. In case it is by the adversary $\mathcal{A}$, this value would not be legitimate as $\mathcal{A}$ has no knowledge of CK' (since the participant's device has not been corrupted) nor CK (a random string which has never been sent), unless $\mathcal{A}$ is able to guess the value of either CK' or CK which is bounded by $q_H/2^{l_h}$. In case the hash function has been queried before by the participant, it means that it was generated in a previous session and is now being used for a replay attack. In this case $\mathcal{A}$ needs to produce a legitimate V_i. The $\mathcal{A}$ can produce legitimate V_i with a chance of at most $q_H/2^{l_h}$.

2. **Case 2** In this case the $\mathcal{A}$ decides to change P_3 and send it back to D_i. Unless the values of P_1 and P_2 are repeated (the probability of this being less than $(q_s + q_e)/2^{l_h}$), P_4 would never have been queried by the participant before. In this case $\mathcal{A}$ would need to produce a legitimate P_4 by constructing A_i'. The probability of this is less than $q_H/2^{l_h}$
3. **Case 3**. In this case $\mathcal{A}$ does not change any P_i. Unless $H(P_3||N_1 \times N_2 \times abG)$ has been queried before, $\mathcal{A}$ would be not be able to learn the SK returned in the **Test** (since this is a random string). However, let us consider a situation where this query has been executed before. We denote this event as E_{query}. In such a case, the simulator can extract the input $N_1 \times N_2 \times abG$ from the hash list and multiply it by $(N_1 \times N_2)^{-1}$ to obtain the CDH result. If E_{query} has not happened before, we can see the guess is not relevant to the key anymore and the $\mathcal{A}$ will have a 50/50 chance of correctly guessing this value.

Thus

$$\Pr(Win_3|\textbf{Case 1}) \leq \frac{2q_H}{2^{l_h}} \tag{A.4}$$

$$\Pr(Win_3|\textbf{Case 2}) \leq \frac{q_s + q_e + q_H}{2^{l_h}} \tag{A.5}$$

$$\Pr(Win_3|\textbf{Case 3} \cap E_{query}) = CDH \tag{A.6}$$

$$\Pr(Win_3|\textbf{Case 3} \cap \overline{E_{query}}) = 1/2 \tag{A.7}$$

Since

$$\begin{aligned}\Pr(Win_3) &= \Pr(Win_3 \cap \textbf{Case 1}) + \Pr(Win_3 \cap \textbf{Case 2}) + \Pr(Win_3 \cap \textbf{Case 3}) \\ &\leq \Pr(Win_3|\textbf{Case 1}) + \Pr(Win_3|\textbf{Case 2}) + \Pr(Win_3|\textbf{Case 3}) \\ &\leq \frac{q_s + q_e + 3q_H}{2^{l_h}} + 1/2 + CDH \end{aligned} \tag{A.8}$$

By combining the equations Eq. A.1, A.2, A.3, A.8, we have

$$\Pr(Win(\mathcal{A}, \mathcal{P})) \leq \frac{{q_H}^2 + (q_s + q_e)^2}{2^{l_H}} + \frac{q_h + q_e + q_s + 3q_H}{2^{l_h}} + 1/2 + CDH \tag{A.9}$$

And this ends the proof. □