electronics

Article

Two-Level-Composite-Hashing Facilitating Highly Efficient Anonymous IoT and D2D Authentication

Hung-Yu Chien

Department of Information Management, National Chi Nan University, Nantou 54561, Taiwan; hychien@ncnu.edu.tw

Abstract: Resource limitation is quite popular in many Internet of Things (IoT) devices and eavesdropping on the identities of IoT devices could reveal the sensitive information; therefore, high efficiency (computation and communication) and anonymity protection are two desirable properties in IoT authentication and in device-to-device (D2D) authentication. Conventionally, dynamic pseudonyms are widely adopted to protect the device identity privacy in IoT authentication and in D2D communications; however, the conventional mechanisms of pseudonym-renewing and pseudonym-bound-public-keys updating could be very costly or be vulnerable to the desynchronization-based denial-of-service (DoS) attacks. In this paper, we propose a novel 2-level composite hashing (2LCH) mechanism to mitigate the problems, and propose the 2LCH-based anonymous IoT and D2D authentication schemes. The schemes simultaneously achieve high efficiency and strong anonymity for such environments; once two devices successfully complete one instance of the server-assist anonymous authentication, they can run several instances of the direct D2D anonymous authentication without the involvement of the server. The merits of the schemes include: (1) high efficiency in terms of computation and communication; (2) easy and efficient generation/synchronization of dynamic pseudonyms; (3) robustness to both desynchronization-based DoS attacks and the unreliable connections; (4) easy application to the existent IoT architectures and standards; and (5) formal security verification.

Keywords: Internet of Things; authentication; MQTT; hash; cloud; edge services

Citation: Chien, H.-Y. Two-Level-Composite-Hashing Facilitating Highly Efficient Anonymous IoT and D2D Authentication. *Electronics* **2021**, *10*, 789. https://doi.org/10.3390/electronics10070789

Academic Editor: Gyu Myoung Lee

Received: 9 February 2021
Accepted: 22 March 2021
Published: 26 March 2021

1. Introduction

The Internet of Things (IoT) technologies facilitate many potential applications and business opportunities. However, it also incurs the great opportunities for hacking into the systems, when the devices are resource-limited and deployed in unprotected environments. The well-known Mirai malware [1] is just an alarming event to signal the coming pervasive threats on IoT systems.

An authenticated key agreement is an essential security mechanism for guarding the security of IoT systems. However, by collecting the devices' identities during the authentication process, attackers might derive the sensitive information, especially when the devices are closely attached to the persons or deployed in some critical facilities like houses, power plants, or vehicles [2,3]. Many IoT devices own limited resources in terms of computing capacities, batteries, or storage space. Therefore, anonymous IoT authentication schemes and D2D authentication that achieve high efficiency and anonymity protection are crucial to IoT applications. In a weaker anonymity, a sender's real identity is only seen by the intended receivers. In a stronger anonymity, any attackers cannot link the transmissions from the same sender.

In this paper, we will propose the 2-level composite hashing (2LCH) and 2LCH-based anonymous IoT/D2D authentication schemes. The schemes achieve high efficiency in terms of computation and communication, and supports both identity anonymity and

unlinkability. We verify the security properties (mutual authentication, session key privacy, and forward secrecy) using the Automated Validation of Internet Security Protocols and Applications (AVISPA) [3], which is one formal security protocol verification toolset. The contributions of this paper are five-fold. (1) Proposal of the 2LCH to facilitate anonymous and efficient identification; (2) new anonymous IoT/D2D authentications with high efficiency; (3) the robustness of the schemes to both desynchronization-based DoS attacks and unreliable connections; (4) easy application of the proposed schemes in many classic IoT architectures and standard IoT protocols like Message Queue Telemetry Transport (MQTT) [4,5]; and (5) formal security protocol verification using AVISPA.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces our system architecture, and proposes our schemes. Section 4 introduces the application of our proposed schemes in the MQTT 5.0 context. Section 5 analyzes the security and evaluates the performance. Section 6 states our conclusions.

2. Related Work

Several publications like [2,6] have examined the vulnerabilities of IoT systems and have surveyed the existent authentication mechanisms. Regarding the cryptographic algorithms of IoT authentication, they could be either asymmetric-key-cryptographies or symmetric-key-cryptographies. In general, a symmetric-key-based scheme demands fewer computations than an asymmetric-key-based scheme. As many IoT devices are resource-limited, and intensive computations and frequent connections would consume large resources and quickly deplete the battery, it is desirable to have very lightweight solutions for authenticating the devices and protecting their identities.

IoT systems range from small-scale ones to very-large-scale systems. For those widely deployed IoT systems, it is cost-effective to choose public mobile systems like 4G/5G as the IoT communication backbone to leverage the advantages of the ubiquitous coverage, high reliability, and very competitive cost [5,6]. The Third Generation Partnership Project (3GPP) has standardized the D2D communication as one critical technology and application domain for the 5G systems. Potential applications of D2D communications include social network services, gaming, information sharing, advertising, conferencing services, etc. Since then, several works like [7–12] aim at proposing IoT authentication schemes and the works like [13–22] aim at securing D2D authentications in the 5G architecture.

Regarding mobile-system-based IoT authentications, the aggregated communication overhead and the long latency would seriously degrade the performance; therefore, reducing the two metrics are important. In these scenarios, both cloud-based architectures or edge-service-based architecture are popular. Morabito et al. [23] designed a lightweight edge computation for wide-range IoT applications. Li et al. [24] proposed to adopt edge computation to improve the latencies and the response time of cloud-based IoT services. In Chien's works [9,25], the systems distribute some tokens to the visited domains for authentication delegation and delicately apply hash functions to greatly improve the computational efficiency; however, the systems do not provide device anonymity and do not provide D2D communications. Fang et al. [12] consider an interesting mobile-system-based IoT architecture, where there are two levels of IoT devices; the level-1 IoT devices are registered at the servers while the level-2 devices are registered at the level-1 IoT devices, which act as a relay for the level-2 devices to access the core networks; the level-2 devices use the same pseudonyms in the communications, which facilitates attackers link the transmissions from the same device. Mahapatra et al. [21] considered the D2D authentication via the help of servers; however, they do not consider device identity privacy. Sun et al. [22] consider both the one-to-one D2D authentication and the one-to-many D2D authentication; however, the schemes use the identity-based prefix encryption (IDPE) [26], which involves several costly pairing operations to protect a device' s identity privacy.

As the potential applications of D2D communications cover a wide range, the corresponding security requirements also vary a lot. Even so, most of secure D2D communications highlight the importance of device anonymity protection and efficiency, as a person might carry the devices or the devices might be deployed at some sensitive locations. To achieve anonymity, many of the existent authentication protocols adopt pseudonyms, and many of them link pseudonyms to some kinds of public-key certificates (PKC) [16] or certificateless public keys (CLPK) [12–14,20]. Even though this approach protects against attackers from getting the real identities, it should frequently update the pseudonyms and the related public keys/CLPKs; otherwise, by correlating the repeatedly used pseudonyms in the transmissions, it could be used to link the transmissions from the same device or even derive sensitive information. Unfortunately, renewing pseudonym-bound certificates/CLPKs with their trusted authority (TA)/key generating center (KGC) is very costly. This weakness causes the application of the approach in a serious dilemma. Therefore, we would like to design a novel pseudonym renewing mechanism that is automatic and highly efficient.

As anonymity/unlinkability is one important focus of this work, we would further elaborate more details of the existing approaches of anonymity protection as follows.

(1) Updating-and-synchronizing pseudonym. The challenges of adopting pseudonyms to achieve anonymity include (a) how do we efficiently maintain the pool of pseudonyms, and (b) how do we synchronize the pseudonyms to avoid possible desynchronization-based DoS attacks. Preloading lots of pseudonyms are suggested in some publications; however, it is not suitable for those resource-limited devices. Other systems update and synchronize the pseudonym during each authentication session. Updating-and-synchronizing pseudonyms per session does not require large storage space, but it should tackle the threats of possible desynchronization attacks or it incurs very costly overhead to renew the corresponding certificates or CLPKs.
(2) Group signature [27] and ring signature [28]. A group signature facilitates one member to sign a signature on behalf of its group without disclosing its identity. A ring signature enables the signer to generate signatures embedded with all members' identities. These two mechanisms can hide the signer's identity under the umbrella of a group of members, even to the authenticators; however, they all demand heavy computations, and have poor scalability and poor dynamic membership management.
(3) Probabilistic encryption. Applying an IND-CCA2-secure encryption [29] on the identities could conceal the identities. The limitation of this approach is its heavy computations, and it does not fit resource-limited devices.
(4) Double Hashing: Ohkubo et al. [30] used two hashing functions to protect untraceability: the first-level hashing is to update the seed, and another hash function is applied on the seed value to have a random pseudonym. Unfortunately, the scheme needs to perform exhaustive search, and replay attacks are not tackled.
(5) Error correction codes (ECC) with secret parameters: Encoding the identities using a code word with some random errors can securely deliver the message to the entities who own the secret generator matrix. The schemes like [31–34] adopted this approach. However, the schemes either suffer from some security attacks [31] or demand costly computations that low-end devices cannot afford [32]. The schemes [33,34] have solved the previous weaknesses, but their scalability is limited.
(6) Encryption of identities using an ephemeral Diffie–Hellman (D–H) key: In this approach, two parties try to establish an ephemeral Diffie–Hellman (D–H) key, and then encrypt the identities, using the key. The authentication between the level-1 devices and the server in [12] adopts this approach. This approach is promising as long as the device has the capacities to hold all the CLPKs or PKCs of the potential receivers.

In [35], Chien proposed a 1-Level composite hashing (1LCH); based on 1LCH, Chien proposed an anonymous D2S protocol, and an anonymous server-assist device-to-server-

to-device (D2S2D) protocol; however, the ordinary hash function is used as the message authentication code in the scheme, which might result in forgery attacks. This paper corrects the weakness, and greatly extends the previous work [35] in several respects. First, we propose a novel hashing and identification mechanism called 2LCH, which could efficiently generate and synchronize pseudonyms. Second, we add two new anonymous protocols called the server-assist-once-a-while D2S2D authentication protocols and the direct D2D authentication protocol. Third, compared to the previous work [35,36], the new protocols are much more efficient; and we facilitate direct D2D anonymous authentication. Fourth, we introduce how to implement our protocols using the MQTT 5.0 AUTH API. Fifth, we verify the security properties in AVISPA.

Both AVISPA and the Burrows–Abadi–Needham (BAN) logic [37] are two popular tools for verifying security protocols; however, there are several reports like [20] showing the limitations of the BAN logic and the failure in verifying the claimed goals. Contrarily, AVISPA, even though it has some limitations like its limited algebra expressions, provides several tools like OFMC (On-the-Fly-Model-Checker) [38] to plot possible attacks on the specification to rigidly check the safety of the specification. Here, we adopt AVISPA for verification. Users specify their security protocols using the High-Level Protocol Specification Language (HLPSL) [39]. AVISPA can simulate the execution flow of a protocol or attack a specified protocol. Embedded tools like OFMC try to plot possible attacks on the specifications. If it successfully plots any attacks, then it will report the weaknesses and the attack process; otherwise, it reports "SAFE". In addition to the above discussed features and security properties. One less-discussed feature in the existent publications is whether a proposed IoT authentication scheme could be easily applied on the standard IoT communication protocols. MQTT is one popular IoT communication protocol; it is lightweight, very easy to use, and widely deployed globally. To gain its lightweight, MQTT avoids the supporting of integrity protection and confidentiality; instead, it assumes a secure sockets layer (SSL) is being supported underlying a MQTT deployment. There exist many MQTT security-enhanced proposals. Some of them designed the specialized hardware to reduce the SSL computation overhead on IoT devices. Some proposals like [40] propose authenticated key agreement schemes for MQTT systems. Chien et al. [41] proposed a two-phase authentication framework in which any secure key agreement scheme can be performed in the first phase, and then the hash value of the session key is supplied as a password in the second phase MQTT CONNECT API. Chien [35] proposed an anonymous IoT protocol in the MQTT 3.11 CONNECT API. In this paper, we will show how to implement our D2S protocol, our D2S2D protocol using the MQTT 5.0 Auth API [5].

3. The Proposed Schemes

In this section, we proposed the 2LCH and the 2LCH-based anonymous IoT and D2D authentication schemes. We first introduce the system architecture. As our schemes can be applied in any IoT architectures where there are servers (denoted as *S*, which could be either cloud servers or edge servers) responsible for managing the devices (denoted as *D*); here we avoided using the specific terms from 5G to avoid the possible misconception that the schemes are limited to the 5G domain only. Table 1 lists the notations. The channels between devices and servers and among devices are susceptible to various attacks. This paper focused on the authentications between devices and on the authentications between a device and its server. The servers are trusted, but the devices might be compromised.

The schemes of this work include three parts. (1) Device-to-server (D2S) anonymous authentication. (2) Server-assist (S-assist) device-to-device (D2D) anonymous authentication, which facilitates anonymous device-to-device authentication via the help of servers. (3) Direct D2D anonymous authentication in which two devices directly authenticate each other. With the authentications, the authenticated entities can share the secure session keys in their transmissions. Each scheme consists of two phases: the registration–initialization phase and the respective authentication phase.

3.1. The Initialization Phase and Two-Level Composite Hashing

All the D2S authentication scheme, the S-assist D2D authentication scheme, and the direct D2D authentication depend on the proposed 2LCH, which can facilitate the anonymous device identification.

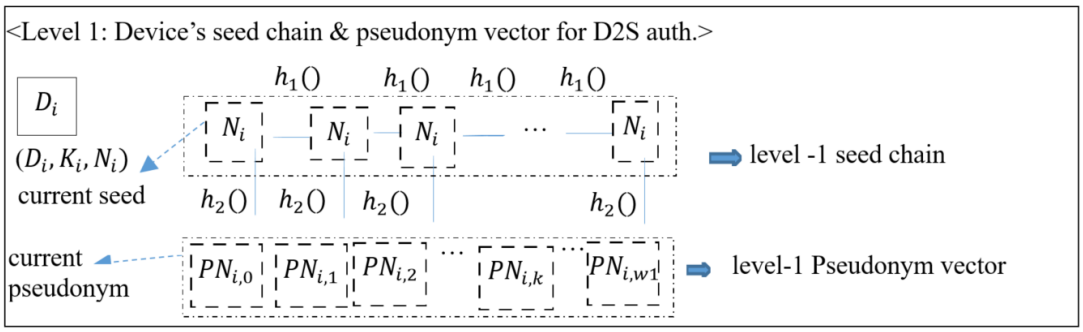
3.1.1. Two-Level Composite Hashing

Figure 1 shows the proposed 2LCH. The level-1 seed and pseudonyms are used for the D2S authentications, and the level-2 seed and pseudonyms will be used in the S-assisted-once-a-while D2S2D authentication and in the direct D2D authentication. Figure 1a shows D_i's level-1 seed and pseudonyms; D_i repetitively applies $h_1()$ on its seed N_i to update N_i, and each active N_i is applied $h_2()$ to have the active pseudonym $h_2(N_i)$.

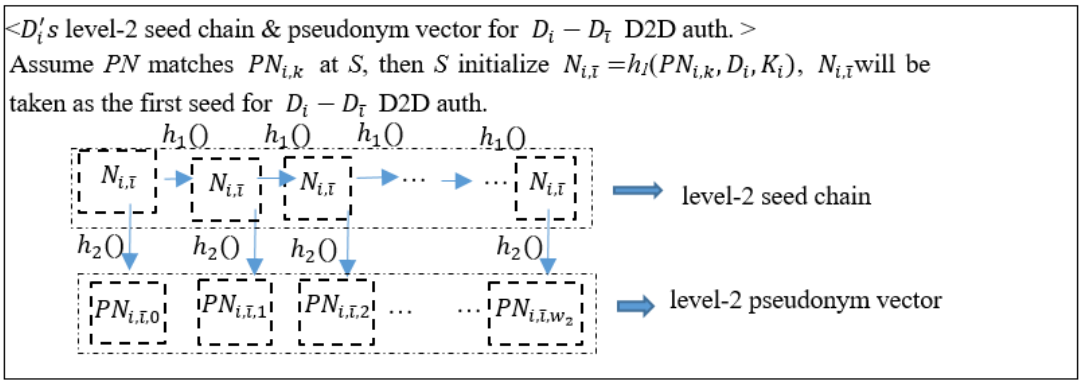
Table 1. The notations.

D; S.	**Device; Server.**
$h_1()$, $h_2()$ *Mac*()	Two cryptographic hashing functions. *Mac*(): message authentication code function, which could be implemented, using HMAC [42].
D_i, $D_{\bar{i}}$; K_i	D_i: ith device; $D_{\bar{i}}$: $\bar{i}$th device; K_i: the secret key shared between D_i and *S*.
N_i	N_i: $D_i's$ *level-1 seed* (a random value) for generating the chain of seed values. Each seed value is used to generate the corresponding level-1 pseudonym $PN_{i,k}$.
${h_1}^j(N_i)$	${h_1}^j(N_i) = h_1(h_1(\ldots h_1(N_i)))$ means h_1 being applied j times.
$PN_{i,0}$, $PN_{i,k}$	$PN_{i,0}$ denotes $D_i's$ $level-1$ $pseunonym$ $PN_{i,0} = h_2(N_i). PN_{i,k} = h_2\left({h_1}^k(N_i)\right)$ denotes the kth $pseunonym$ successorof $PN_{i,0}$
PV, PV_i	PV: $Pseudonym\ Vector$; $PV_i = \{PN_{i,0}, PN_{i,1}, \ldots, PN_{i,k}, \ldots, PN_{i,w1}\}$ is the level-1 pseudonym vectors pre-calculated by *S* for D_i.
w_1, w_2	w_1, w_2: two values respectively specify the window sizes of the precalculated level-1 and level-2 pseudonym vectors.
$N_{i,\bar{i}}$	$N_{i,\bar{i}} = h_1(PN_{i,k}, D_i, K_i)$ is the first seed for the level-2 pseudonym chain of D_i, where D_i is the $D_i - D_{\bar{i}}$ D2D authentication initiator, $D_{\bar{i}}$ is the responder, and $PN_{i,k}$ is the matched pseudonym in the level-1 pseudonym vectors.
$PN_{i,\bar{i},j}$	$PN_{i,\bar{i},j} = h_2(h_1^j(N_{i,\bar{i}}))$ for $j > 0$, where $N_{i,\bar{i}}$ is $D_i's$ current active seed for level-2 $D_i - D_{\bar{i}}$ D2D authentication.
$PV_{i,\bar{i}}$, $PV_{\bar{i},i}$	$PV_{i,\bar{i}} \coloneqq \{PN_{i,\bar{i},0}, PN_{i,\bar{i},1}, \ldots, PN_{i,\bar{i},w2}\}$ is the precalculated level-2 pseudonym vectors for identifying D_i by $D_{\bar{i}}$. Likewise, $PV_{\bar{i},i}$ denotes that vector for D_i to identify $D_{\bar{i}}$.
$Enc_{K_i}()$	Encryption using the key K_i.
G_{ECC}, *P*, *q*	G_{ECC} is a cyclic multiplicative group of an order q, where the Computational Diffie–Hellman Problem (CDHP) is hard; *P* is a generator of G_{ECC}. Here, we let G_{ECC} be a group in the elliptic curve setting for the short key size.

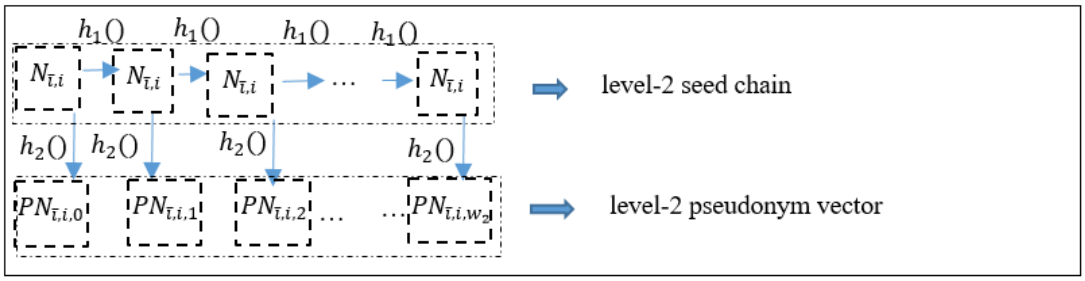
Figure 1b shows D_i's level-2 seed and pseudonyms for $D_i - D_{\bar{i}}$ D2D authentication, where $D_{\bar{i}}$ is D_i's communicating partner in the authentication. Assume $PN_{i,k}$ is D_i' active pseudonym, then $D_i's$ level-2 seed is set as $N_{i,\bar{i}} = h_1(PN_{i,k}, D_i, K_i)$ for the $D_i - D_{\bar{i}}$ D2D authentication. Like the level-1 operation, $h_1()$is repetitively applied on $N_{i,\bar{i}}$ to update its values, and $h_2()$ is applied on the active $N_{i,\bar{i}}$ to have the active pseudonym $h_2(N_{i,\bar{i}})$ for the $D_i - D_{\bar{i}}$ D2D authentication. Likewise, Figure 1c shows that for $D_{\bar{i}}$'s $D_{\bar{i}} - D_i$ D2D authentication. Figure 2 sorts out the algorithms for the entities.



(**a**)



(**b**)



(**c**)

Figure 1. (**a**) D_i's level-1 seed and pseudonyms; (**b**) D_i's level-2 seed and pseudonyms for $D_i - D_{\bar{i}}$ D2D authentication; (**c**) $D_{\bar{i}}$'s level-2 seed and pseudonyms for $D_{\bar{i}} - D_i$ D2D authentication.

We now introduced the registration–initialization phase using the 2LCH.

3.1.2. The Registration and Initialization Phase

The registration is conducted in a secure channel. Each device D_i registers its identity D_i (here we used the same notation to denote both the entity and its identity), the secret key K_i, and its seed N_i (for pseudonym generation) at the S.

N_i is the seed for generating the pseudonyms. When D_i initiates an authentication request, it computes $PN = h_2(N_i)$ as the current pseudonym and updates $N_i = h_1(N_i)$ for the next session. This is formulated as the algorithm PN_Seed_Update(N_i) called by D_i in Figure 2.

When S accepts $D_i's$ registration (D_i, N_i, K_i), it initializes $D_i's$ pseudonym vectors $PV_i = \{PN_{i,0}, PN_{i,1}, \ldots, PN_{i,k}, \ldots, PN_{i,w1}\}$, where $PN_{i,j} = h_2\left({h_1}^j(N_i)\right), j = 0{\sim}w1$. This is depicted in Figure 1a and in Figure 2. This arrangement of preparing the vectors of one current pseudonym and the future w_1 pseudonyms is for deterring possible desynchronization-based DoS attacks and for handling the unreliable connection issues. When either desynchronization-based DoS attacks or unreliable connections happen, the requests from D_i might be blocked or be lost, and D_i would repetitively retry its requests. In such situations, S who prepares additional w_1 pseudonyms can properly tackle the issues. This initialization of $D_i's$ pseudonym vectors is formulated as the algorithm PV_Iinitialize(D_i, N_i) called by S in Figure 2.

Device_init(D_i) is for initializing $D_i's$ initial parameters. $D_i's$ PN_Seed_Update(N_i) method is executed when each time D_i initiates a D2S authentication request; in this method, D_i first generates the current pseudonym $PN = h_2(N_i)$ and then updates its seed $N_i = h_1(N_i)$.

<<Device's methods and S's methods for D2S authentication>>

D_i

1. Device_init(D_i)[$D_i's$ initialization]
   $\{(D_i, K_i, N_i)\}$
2. [$D_i's$ internal state during D2S auth.]
   $(D_i, K_i, N_i, PV_i = \{PN_{i,k-w1}, \ldots, PN_{i,k}\})$, where $PN_{i,k}$ is the current active pseudonym and $\{PN_{i,k-w1}, \ldots, PN_{i,k-1}\}$ is the window of past *w1 PNs*. Keeping a window of *w1*-past PNs is for resisting possible desynchronization-based DoS attacks or unreliable connections.
3. PN_Seed_Update(N_i) [D_i' internal state update when it launches an conn.]
   Generate pseudonym $PN = h_2(N_i)$, Update *Seed* $N_i = h_1(N_i)$}

S

1. PV_Iinitialize(D_i, N_i)[Initialize Device's pseudonym vector]
   {For each D_i and its active N_i
   Compute $PN_{i,j} = h_2\left({h_1}^j(N_i)\right), j = 0{\sim}w1$, where ${h_1}^j(N_i) = h_1(h_1(\ldots h_1(N_i)))$ means h_1 being applied j times. Pseudonym Vector (PV) for D_i: $PV_i = \{PN_{i,0}, PN_{i,1}, \ldots, PN_{i,k}, \ldots, PN_{i,w1}\}$
   }
2. PV_Update(*PN, Req*): [Update pseudonym vector when an incoming *PN* matches an entry]
   Use the incoming *PN* to locate a matched entry (say $PN_{i,k}$) and verify *Req*

Figure 2. D_i's methods and *S*'s methods for the Level-1 D2S authentication.

3.2. The Device-to-Server (D2S) Anonymous Authentication

Now we are ready to introduce the D2S anonymous authentication. Figure 3 shows the authentication flow. Basically, this scheme is my anonymous version of the classic challenge-and-response authentication; D_i sends its current pseudonym $PN = h_2(N_i)$ for *S* to identify the device, and they use the secret key K_i to calculate the responses. The details are described as follows.

Step 1. $D_i \rightarrow S$: PN, ID_S, X, Req

D_i executes PN_Seed_Update(N_i) to generate the pseudonym PN and updates N_i; D_i chooses $x \in_R Z_q^*$, and calculates $X = xP$ and *Req*= $Mac(K_i, PN, X)$ for *S* to validate the request.

Step 2. $S \rightarrow D_i$: $ID_S, PN, Y, Res1$

S executes PV_Update(*PN, Req*), which uses *PN* to search a matched entry in $D_i's$ pseudonym vectors, and to update $D_i's$ pseudonym vectors. *S* chooses $y \in_R Z_q^*$, and calculates $Y = yP$ and its response $Res1 = Mac(K_i, PN, ID_S, X)$ (which is a response to $D_i's$ challenge *PN*). *S* sends the messages specified in Step 2.

Step 3. $D_i \rightarrow S$: $PN, ID_S, Res2$

D_i verifies the received *Res1*, calculates $Res2 = Mac(K_i, PN, ID_S, Y)$, and computes the session key $K_{sess} = h_1("KEY", ID_S, PN, K_i, xyP)$. It sends the data specified in Step 3.

Upon receiving $D_i's$ response, S verifies *Res*2 and computes the session key.

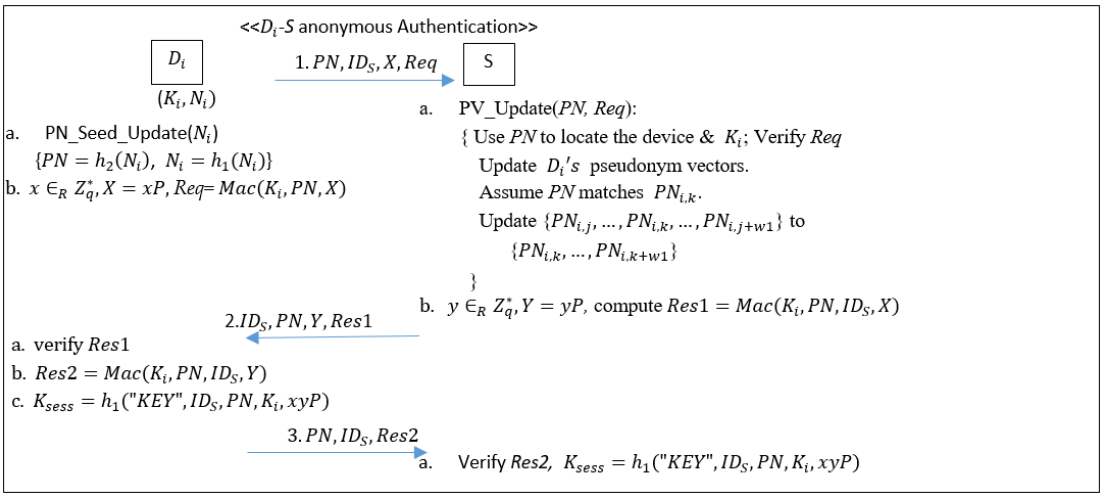


Figure 3. D_i-S's level-1 D2S authentication flow.

3.3. The S-Assist D2D Anonymous Authentication

Based on the D2S anonymous authentication, this section introduces how two devices within the domain of a server can perform D2D anonymous authentication and establish session keys with the help of the server. There are two versions of the S-assist D2D anonymous authentication. The first version requires the help of the server each time when two devices execute the D2D anonymous authentication; this version is referred as the S-assist-always device-to-server-to-device (D2S2D) anonymous authentication. The second version facilitates two devices perform $w2$ instances of their direct D2D anonymous authentication without the involvement of the server once they have successfully run one instance of the S-assist-once-a-while D2S2D anonymous authentication.

3.3.1. The S-Assist-Always D2S2D Anonymous Authentication

This version needs only the first level of device's seed-pseudonym structure (specified in Figure 1a). Figure 4 shows the flow, where D_i and $D_{\bar{i}}$ are the two devices executing the mutual anonymous authentication. Basically, this is an anonymous version of the classic challenge-and-response authentication among three entities (two devices and one server) where the server helps to distribute a temporary secret key $K_{tmp,i,\bar{i}}$ to both D_i and $D_{\bar{i}}$ so that they can authenticate each other. The details are described as follows.

Step 1. $D_i \rightarrow S$: $PN, ID_S, X,\ Req$

D_i executes PN_Seed_Update(N_i) to generate the pseudonym PN and updates N_i; D_i also calculates $Req = Enc_{K_i}(D_{\bar{i}}, PN)$ for S to decrypt the request to derive the identity $D_{\bar{i}}$.

Step 2. $S \rightarrow D_{\bar{i}}$:. $ID_S, PN_{\bar{i},\bar{k}}, PN, R1, E_2, Res2, E_3, Res3$

S first launches PV_Update(*PN, Req*) to identify the device, gets the secret key K_i, and decrypts/validates *Req*. If the verification succeeds, then it updates $PV_i = \{PN_{i,k+1}, \dots, PN_{i,k+1+w}\}$.

S then chooses a random challenge *R1*, computes the temporary secret key $K_{tmp,i,\bar{i}} = h_1("Server", ID_S,\ , PN, D_{\bar{i}}, K_i, R1)$ for D_i and $D_{\bar{i}}$, and prepares the encryption $E_2 = Enc_{K_{\bar{i}}}(D_i, K_{tmp,i,\bar{i}}, X)$ and $Res2 = Mac(K_{\bar{i}}, E_2, R1)$ for $D_{\bar{i}}$.

S locates $D_{\bar{i}}'s$ active PN (say $PN_{\bar{i},\bar{k}}$), updates $D_{\bar{i}}'s$ level-1 pseudonym vectors, and prepares $E_3 = Enc_{K_{\bar{i}}}(D_{\bar{i}}, K_{tmp,i,\bar{i}})$ and $Res3 = Mac(K_i, PN, PN_{\bar{i},\bar{k}}, ID_S, E_3,\ R1)$. It finally sends the data in Step 2 to $D_{\bar{i}}$.

Step 3. $D_{\bar{\imath}} \rightarrow D_i$: $ID_S, PN, PN_{\bar{\imath},\bar{k}}, R1, E_3, Res3, Res4, Y$

$D_{\bar{\imath}}$ decrypts E_2, and gets $K_{tmp,i,\bar{\imath}}$ and X. It validates $Res2$; if the validation succeeds, then it updates $D_{\bar{\imath}}'s$ seed and pseudonym vectors. It chooses a random challenge $y \in_R Z_q^*$, and computes $Y = yP$. It then prepares $Res4 = Mac(K_{tmp,i,\bar{\imath}}, E_3, PN, Y)$, and computes the session key $K_{sess} = h_1("D2D", ID_S, PN, PN_{\bar{\imath},\bar{k}}, K_{tmp,i,\bar{\imath}}, xyP)$, and sends the data specified in Step 3 to D_i.

Step 4. $D_i \rightarrow D_{\bar{\imath}}$: $Res5$

D_i decrypts E_3, gets $D_{\bar{\imath}}$ and $K_{tmp,i,\bar{\imath}}$, computes its own $K_{tmp,i,\bar{\imath}}$, and verifies the received one. It further verifies $Res3$ and $Res4$. If the verifications succeed, it accepts the session, computes the session key K_{sess}, and sends the response $Res5 = Mac(K_{tmp,i,\bar{\imath}}, Y)$.

Step 5. $D_{\bar{\imath}}$: Verify the received $Res5$. If it succeeds, then it accepts the session key K_{sess}.

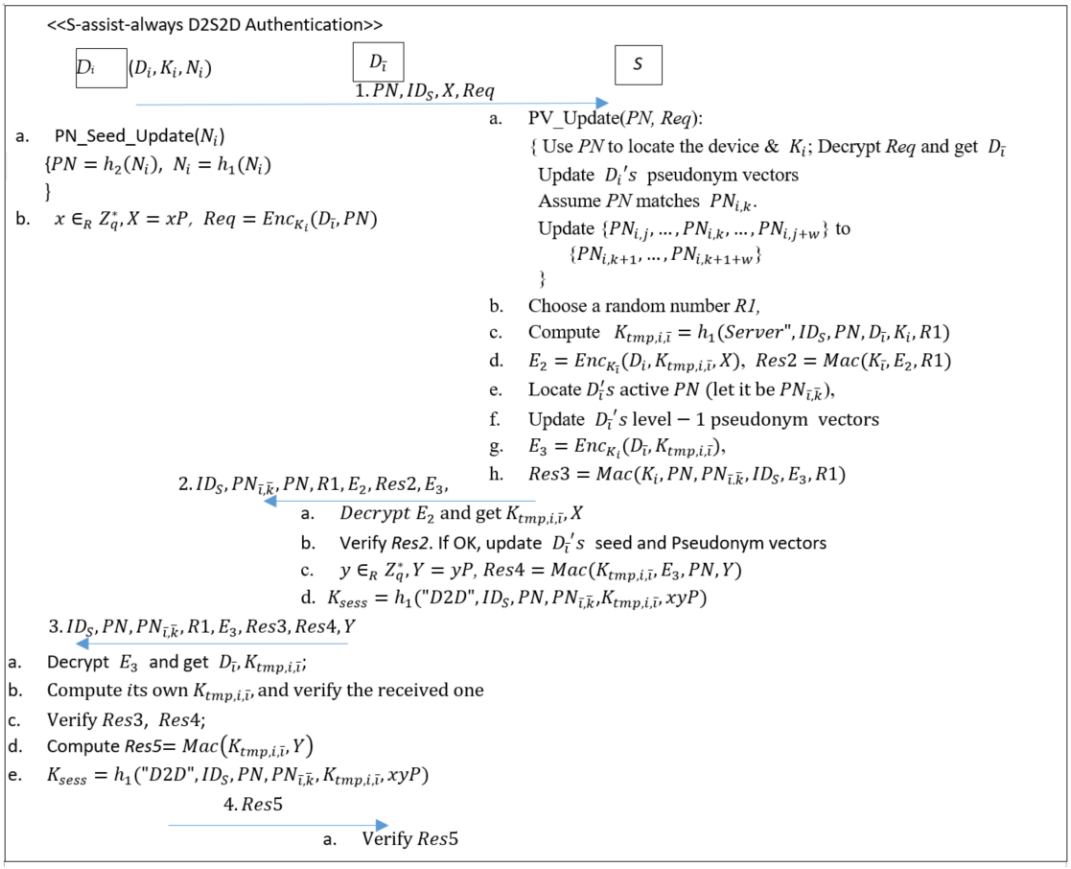


Figure 4. S-assist-always D2D anonymous authentication flow.

3.3.2. The S-Assist-Once-A-While D2S2D Anonymous Authentication and Direct D2D Authentication

This S-assist-once-a-while D2S2D anonymous authentication facilitates two devices run several direct D2D anonymous authentication sessions, which do not require the involvement of servers after they have completed one S-assist-once-a-while D2S2D authentication via the help of a server. Two devices (say D_i and $D_{\bar{\imath}}$) first run the S-assist-once-a-while D2S2D anonymous authentication to establish the shared key $K_{tmp,i,\bar{\imath}}$ and to establish their level-2 seed-pseudonym vectors (as specified in Figure 1b,c); after that, they can run $w2$ instances of the direct D2D anonymous authentication.

The S-Assist-Once-A-While D2S2D Anonymous Authentication

Figure 5 shows the flow of the S-assist-once-a-while D2S2D anonymous authentication. In this instance, we took two devices (say D_i and $D_{\bar{i}}$) as the example to demonstrate how the protocol works. The details are described as follows.

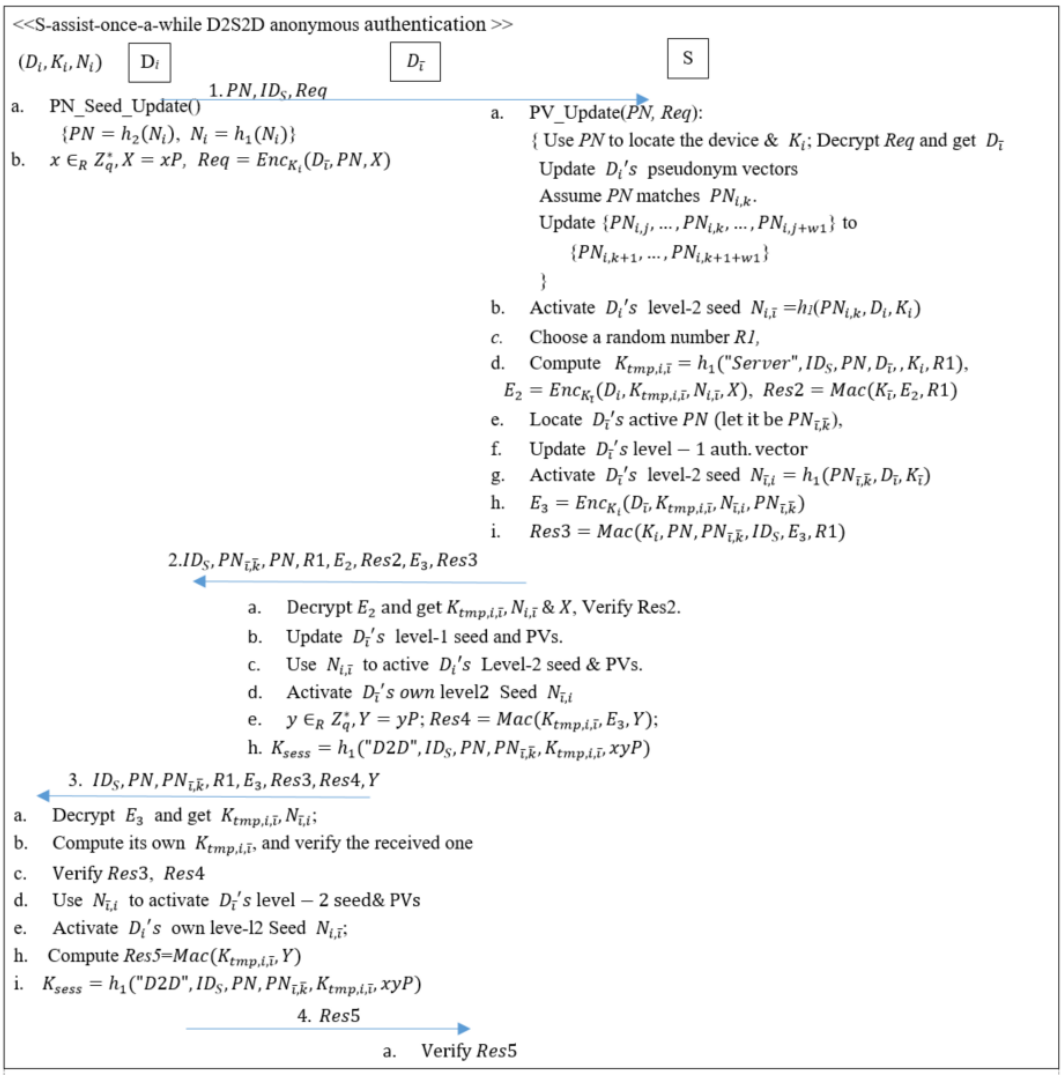


Figure 5. S-assist-once-a-while D2S2D anonymous authentication.

Step 1. $D_i \rightarrow S$: PN, ID_S, Req

D_i executes PN_Seed_Update(N_i) to generate the pseudonym PN and updates N_i; D_i chooses $x \in_R Z_q^*$, and calculates $X = xP$ and $Req = Enc_{K_i}(D_{\bar{i}}, PN, X)$ for S to decrypt the request to derive the identity $D_{\bar{i}}$.

Step 2. $S \rightarrow D_{\bar{i}}$:. $ID_S, PN_{\bar{i},\bar{k}}, PN, R1, E_2, Res2, E_3, Res3$

S first launches PV_Update(*PN, Req*) to identify the identity of the device, gets the secret key K_i, and decrypts/verifies *Req*. If the verification succeeds, then it updates $PV_i = \{PN_{i,k+1}, \dots, PN_{i,k+1+w1}\}$.

S also calculates $D_i's$ level-2 seed $N_{i,\bar{i}} = h_1(PN_{i,k}, D_i, K_i)$, which will be used in the direct D2D anonymous authentication. $D_i's$ level-2 seed-pseudonym vector is depicted in Figure 1b.

S then chooses a random challenge R1, computes the temporary secret key $K_{tmp,i,\bar{i}} = h_1("Server", ID_S, PN, D_{\bar{i}}, K_i, R1)$ for D_i and $D_{\bar{i}}$, and prepares the encryption $E_2 = Enc_{K_{\bar{i}}}(D_i, K_{tmp,i,\bar{i}}, N_{i,\bar{i}}, X)$ and $Res2 = Mac(K_{\bar{i}}, E_2, R1)$ for $D_{\bar{i}}$.

S locates $D_{\bar{\imath}}'s$ active PN (say $PN_{\bar{\imath},\bar{k}}$), updates $D_{\bar{\imath}}'s$ level-1 pseudonym vectors, and activates $D_{\bar{\imath}}'s$ level-2 seed $N_{\bar{\imath},i} = h_1(PN_{\bar{\imath},\bar{k}}, D_{\bar{\imath}}, K_{\bar{\imath}})$; $D_{\bar{\imath}}'s$ level-2 seed-pseudonym vector is depicted in Figure 1c.

S prepares $E_3 = Enc_{K_i}(D_{\bar{\imath}}, K_{tmp,i,\bar{\imath}}, N_{\bar{\imath},i}, PN_{\bar{\imath},\bar{k}})$ and $Res3 = Mac(K_i, PN, PN_{\bar{\imath},\bar{k}}, ID_S, E_3, R1)$. It finally sends the data specified in Step 2 to $D_{\bar{\imath}}$.

Step 3. $D_{\bar{\imath}} \rightarrow D_i$: $ID_S, PN, PN_{\bar{\imath},\bar{k}}, R1, E_3, Res3, Res4, Y$

$D_{\bar{\imath}}$ decrypts E_2, and gets $K_{tmp,i,\bar{\imath}}$ and $N_{i,\bar{\imath}}$. It validates *Res*2; if the validation succeeds, then it updates $D_{\bar{\imath}}'s$ level-1 seed and pseudonym vectors. It also uses $N_{i,\bar{\imath}}$ to activate $D_i's$ level-2 seed-pseudonym vector (as specified in Figure 1b).

It chooses $y \in_R Z_q^*$ and prepares $Y = yP$ and $Res4 = Mac(K_{tmp,i,\bar{\imath}}, E_3, Y)$. It computes the session key $K_{sess} = h_1("D2D", ID_S, PN, PN_{\bar{\imath},\bar{k}}, K_{tmp,i,\bar{\imath}}, xyP)$, and sends the data specified in Step 3 to D_i.

Step 4. $D_i \rightarrow D_{\bar{\imath}}$: *Res*5

D_i decrypts E_3, gets $N_{\bar{\imath},i}, K_{tmp,i,\bar{\imath}}$, computes its own $K_{tmp,i,\bar{\imath}}$, and verifies *Res3*, *Res*4, and the received $K_{tmp,i,\bar{\imath}}$. If the verifications succeed, it uses $N_{\bar{\imath},i}$ to activate $D_{\bar{\imath}}'s$ level2 seed— pseudonym vectors, and activates its own level-2 seed $N_{i,\bar{\imath}}$. It accepts the session and computes the session key K_{sess}. It further sends the response $Res5 = Mac(K_{tmp,i,\bar{\imath}}, Y)$.

Step 5. $D_{\bar{\imath}}$: $D_{\bar{\imath}}$ verifies the received *Res*5. If it succeeds, then it accepts the session key K_{sess}.

The Direct D2D Anonymous Authentication

Once two devices successfully complete one instance of the S-assist-once-a-while D2S2D anonymous authentication, they can run w2 instances of the direct D2D anonymous authentication without the involvement of the server. Figure 6 depicts the flow. The top of Figure 7 shows the internal state of $D_i's$ and $D_{\bar{\imath}}'s$ level-2 seed and pseudonym vectors, and the rest shows the flow. The details are described as follows.

Step 1. $D_i \rightarrow D_{\bar{\imath}}$: $PN, \overline{PN}, X$

D_i executes PN_L2Seed_Update($N_{i,\bar{\imath}}$) to generates its level-2 pseudonym $PN = h_2(N_{i,\bar{\imath}})$ and updates its level-2 seed $N_{i,\bar{\imath}} = h_1(N_{i,\bar{\imath}})$. It chooses a random challenge $x \in_R Z_q^*$, computes $X = xP$, and sends the request $\{PN, \overline{PN}, X\}$, where $\overline{PN}$ is $D_{\bar{\imath}}'s$ active level-2 pseudonym.

Step 2. $D_{\bar{\imath}} \rightarrow D_i$: $PN, \overline{PN}, Y$,*Res2*

$D_{\bar{\imath}}$ runs PV_L2Update(*PN*) to identify the device and to derive the shared key $K_{tmp,i,\bar{\imath}}$. It updates $D_i's$ level-2 pseudonym vectors. It chooses $y \in_R Z_q^*$, and computes $Y = yP$ and $Res2 = Mac(PN, K_{tmp,i,\bar{\imath}}, X, Y)$.

Step 3.$D_i \rightarrow D_{\bar{\imath}}$: $PN, \overline{PN}$,*Res1*

It verifies Res_2. If it succeeds, then it calculates $Res1 = Mac(\overline{PN}, K_{tmp,i,\bar{\imath}}, Y, X)$ and the session key $K_{sess} = h_1("D2D", PN, \overline{PN}, K_{tmp,i,\bar{\imath}}, xyP)$. It updates $D_{\bar{\imath}}'s$ level-2 seed and PV.

Step 4. $D_{\bar{\imath}}$:

$D_{\bar{\imath}}$ verifies *Res*1. If it succeeds, then it computes the session key K_{sess}, and updates its level-2 seed and PVs.

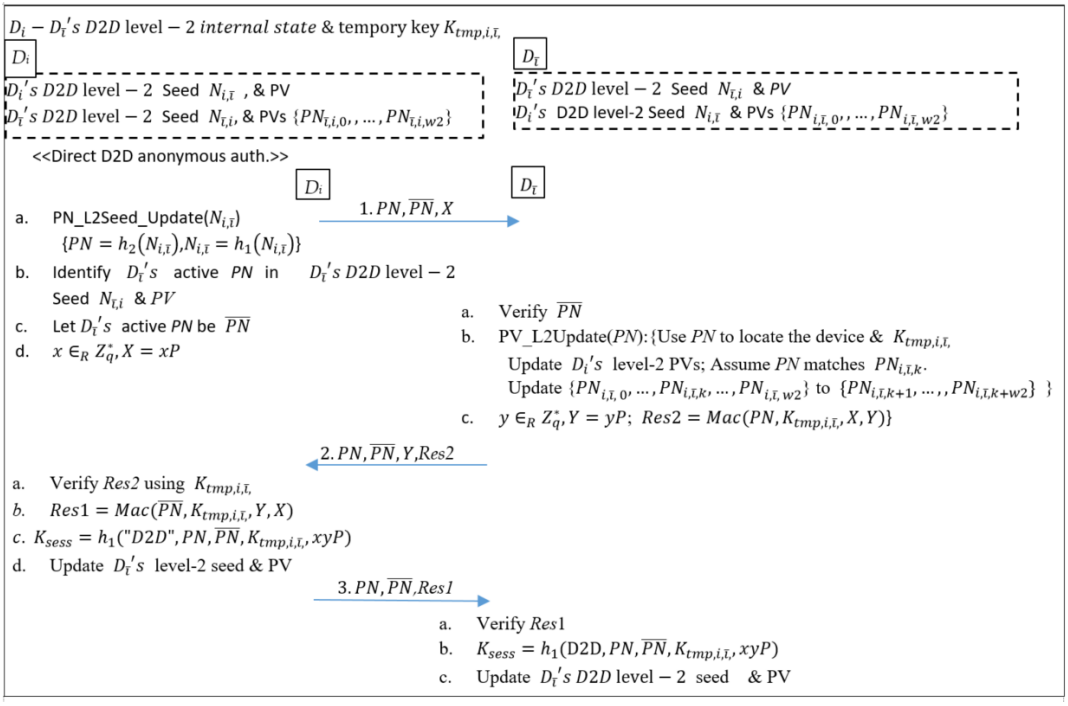


Figure 6. Direct D2D anonymous authentication.

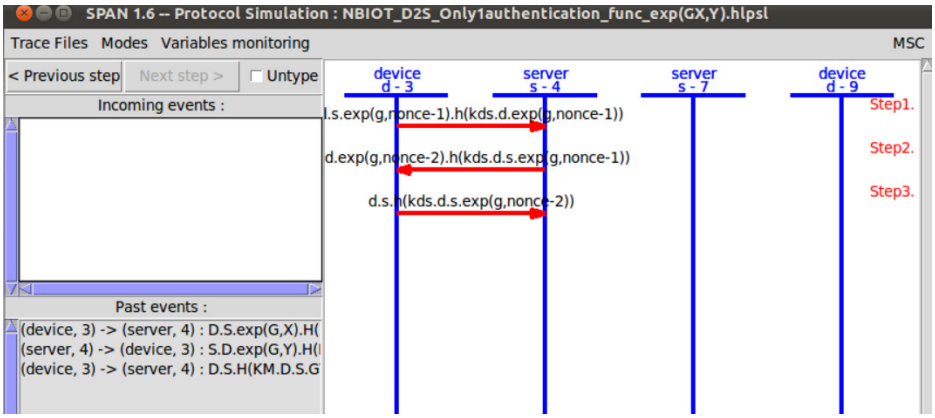


Figure 7. The message sequences of the D2S authentication.

4. Using Our D2S Anonymous Scheme in the MQTT 5.0 Context

In this section, we describe how we can apply our proposed D2S anonymous scheme in the MQTT 5.0 context.

Chien et al. [41] has shown how to implement their two-phase challenge-response protocol for enhancing the MQTT authentication, using the MQTT v3.1 [43] CONNECT API. The new MQTT standards v5.0 [5] has extended the MQTT v.3.1 API to incorporate a new feature called the enhanced authentication. The enhanced authentication facilitates designers implementing their own security protocols, using the new API called AUTH and the extended fields in the existent CONNECT and CONNACK APIs. The enhanced authentication using these APIs facilitates a query-response style message exchange between a MQTT broker and its clients. The client specifies the authentication method field in the CONNECT packet, this field is included through the following AUTH packets exchanged, and it is also specified in the final CONNACK packet.

To implement the proposed protocols, one can define our proposed D2S anonymous scheme as an authentication method (a code), and then specifies the code in the CONNECT/AUTH/CONNACK packets. The exchanged messages in our protocols are transmitted as the authentication data in the exchanged AUTH packets.

5. Security Analysis and Performance Evaluation

5.1. The Formal Protocol Verification in AVSIPA and Security Analysis

In this section, we proved and analyzed the securities in two ways. One is proving the critical security properties like mutual authentication, session key privacy, and forward secrecy in the AVISPA tool set. This approach provides the formal verifications of some critical security properties, but AVISPA does not verify some security properties like anonymity and unlinkability. We, therefore, also analyze all the security properties in Section 5.1.2.

5.1.1. Security Properties Verification in AVISPA

Our schemes consist of four main protocols—anonymous D2S authentication, S-assist-always anonymous D2S2D authentication, S-assist-once-a-while anonymous D2S2D authentication, and direct D2D anonymous authentication. In the HLPSL specification, we define three roles: "server" for the server, "device1" for the device in the D2S authentication and for the initiator in the D2S2D authentication, and "device 2" for the responder in the D2S2D authentication and for the responder in the direct D2D authentication.

AVISPA supports the Dolev-Yao channels, where the intruder may divert sent messages and send new ones impersonating other agents. AVISPA can verify authentication and session key privacy, but it does not support verification of anonymity and unlinkability.

Here, we use it to verify mutual authentication, session key privacy, and forward secrecy. Mutual authentication can be modeled using the predicate "authentication_on", and session key privacy can be modeled using the predicate "secrecy_of". To model the forward secrecy, we let each instance (server and device) run two successive sessions in their respective HLPSL specifications to have two session keys; let the intruder have the secrets of the second session, and then verify whether the intruder can derive the secret keys of the first session. The goal is to verify that, even if the intruder has the secrets of the 2nd session, the goals of authentication and session key privacy of the 1st session are still achieved in AVISPA verification.

Basically, our HLPSL for the four protocols can be classified as two kinds: one kind is the specification of one session with the goals of "authentication_on" of the mutual authentication and "secrecy_of" of the session keys, and the other kind is the specification of two successive sessions with the security goals of the 1st session. The main differences among the specifications of the four protocols include the different shared keys used, the number of entities in the protocols (two parties in the D2S authentication and in the direct D2D authentication, and three parties in the D2S2D authentication). Therefore, in the following, we will introduce and discuss some results, but omit similar others to keep the presentation clear and focused.

Verification on the Two-Party D2S Authentication

Figure 7 shows the protocol simulation of the D2S authentication, where "device d-3" and "server s-4" exchange messages and accomplish the authentication. Figure 8 shows the result of the OFMC verifier on the D2S authentication; the result of the OFMC on the D2S authentication is "SAFE"—no successful attacks can be plotted.

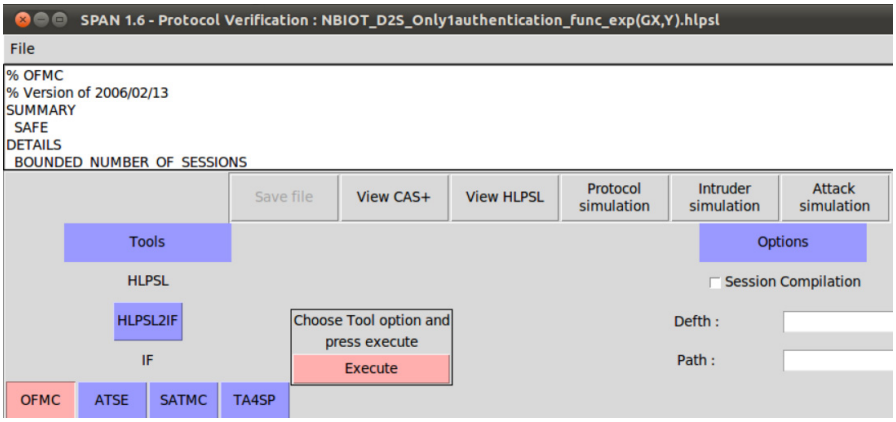


Figure 8. The result of On-the-Fly-Model-Checker (OFMC) on the D2S authentication.

Next, we examined the forward secrecy of the D2S authentication. Figure 9 shows the message flow, where "device d-3" and "server s-4" accomplished two runs of authentications, and the intruder knew the secrets of the 2nd session; the result shows that OFMC verifier could not plot any successful attacks on the 1st session. The result was "SAFE" in Figure 10.

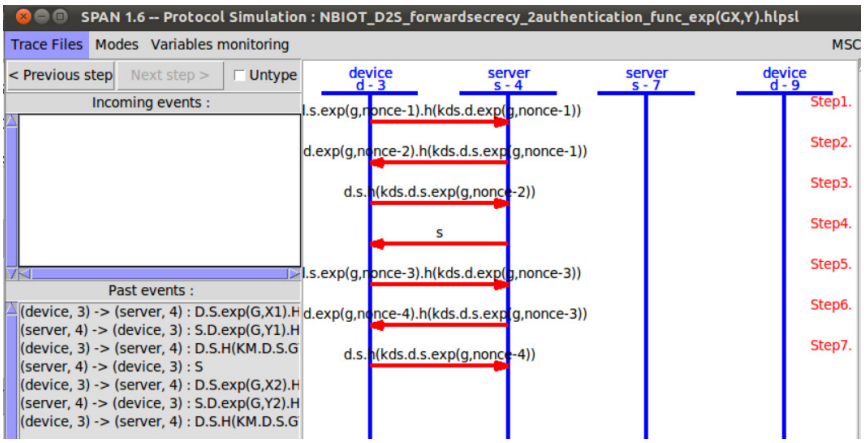


Figure 9. The message sequences of two successive D2S authentications.

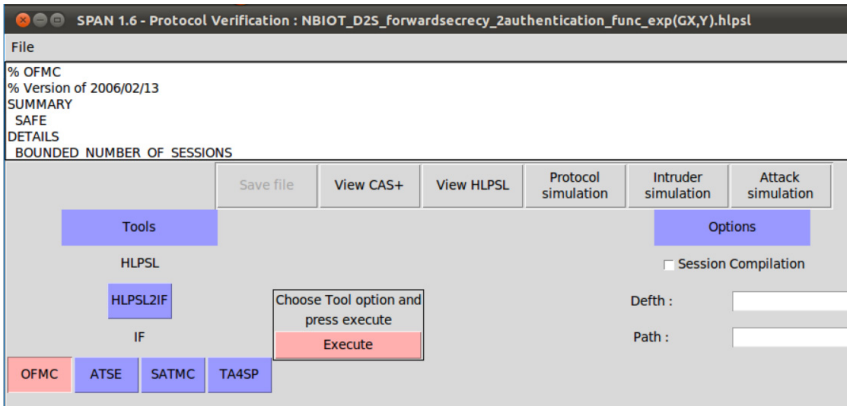


Figure 10. The result of OFMC on two D2S authentication sessions.

Verification on the Three-Party S-Assist-Always D2S2D Authentication

Figure 11 shows the message exchange of the S-assist-always D2S2D authentication, where "device d-3" and "device d-5" accomplished authentication via the help of "server

s-4". OFMC checks "SAFE" on the protocol in Figure 12. The same results could be derived for the forward secrecy of the two-session cases.

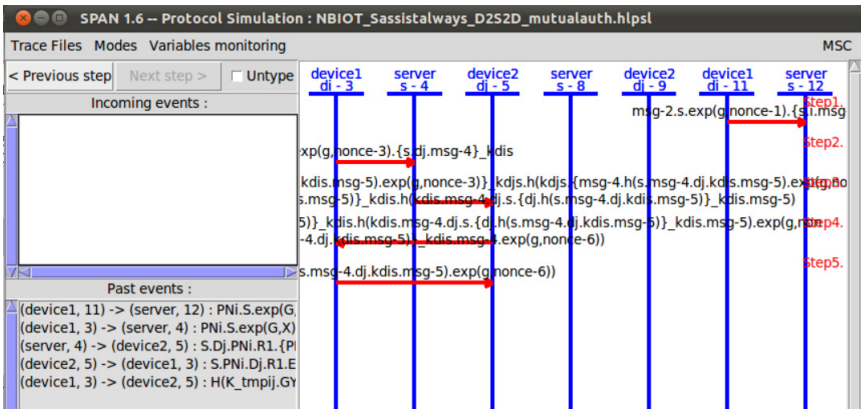


Figure 11. The message exchange of the S-assist-always D2S2D authentication.

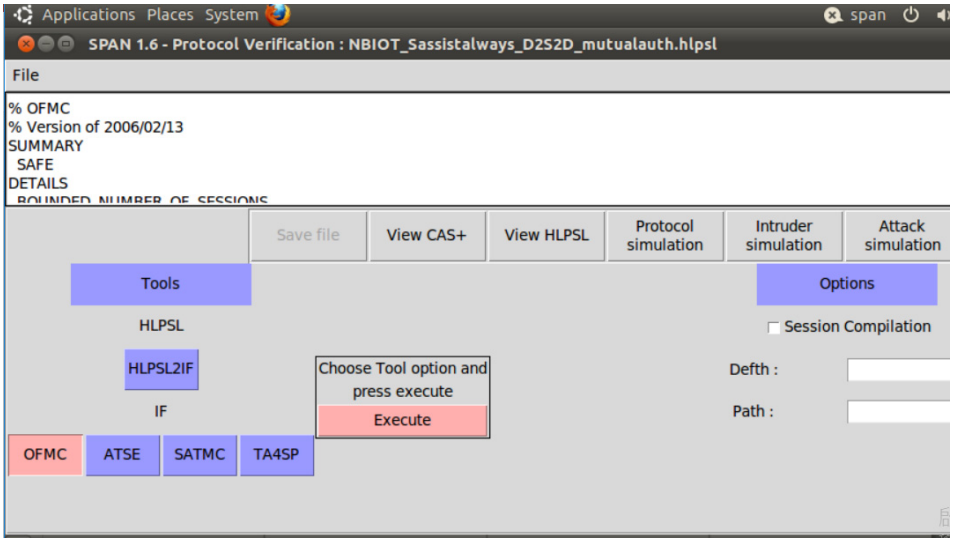


Figure 12. OFMC checks the security of the S-assist-always D2S2D authentication.

The specifications of the S-assist-once-a-while D2S2D authentication were similar to those of the S-assist-always D2S2D authentication, except the shared keys used are different and the identifications (pseudonym) are different. The specifications of the direct D2D authentication are similar to those of the D2S authentication, except that one entity is different, the shared keys are different, and the identifications are different. Here, we omit the simulation pictures to save space and keep the presentation tidy.

5.1.2. Security Properties Analysis

In this section, we analyzed the security properties of the proposed schemes. Table 2 summarizes all the security properties.

Mutual authentication between the initiator and the responder: Mutual authentication between a device and a server in the D2S anonymous authentication; mutual authentication between the two devices in the S-assist-always D2S2D authentication, in the S-assist-once-a-while D2S2D authentication, and in the direct D2D authentication.

From the protocol descriptions, we could see that the initiator D_i always updates its pseudonym PN in each session. This random and fresh PN simultaneously functions as

a pseudonym and as a random challenge to the responders. In all the protocols, the responders (S in the D2S anonymous authentication and $D_{\bar{\imath}}$ in the other protocols) all need to send their responses, which are based on the secret key and the fresh challenge PN. Likewise, the responders also send its own random challenges Y to the initiators (D_i), which are required to send their responses accordingly. These are the classic challenge–response mechanisms. As long as the secret keys are secure and the challenges are random and fresh, this ensures the mutual authentication between the initiator and the responder.

Table 2. Summary of the security properties.

Schemes	Security Properties
D2S authentication	Mutual authentication between a device and a server; anonymity of devices; unlinkability of devices' successful connections; resistance to desynchronization-based DoS attacks; session key forward secrecy.
S-assist-always D2S2D authentication	Mutual authentication between the two communicating devices; Authentication of a server to devices; anonymity of devices; unlinkability of devices' successful connections; resistance to desynchronization-based DoS attacks; session key forward secrecy.
S-assist-once-a-while D2S2D authentication	Mutual authentication between the two communicating devices; authentication Of a server to devices; anonymity of devices; unlinkability of devices' successful connections; resistance to desynchronization-based DoS attacks; session key forward secrecy.
Direct D2D authentication	Mutual authentication between the two communicating devices; anonymity of devices; unlinkability of devices' successful connections; resistance to desynchronization-based DoS attacks; session key forward secrecy.

Authentication of the server to the initiator in the S-assist protocols: In the S-assist-always D2S2D authentication and in the S-assist-once-awhile D2S2D authentication, the initiator (D_i) sends PN as a pseudonym and as a fresh challenge on which the server should generate a proper response, using the secret key and the challenge. This ensures the authentication of the server to the initiator. We also note here that our schemes do not provide the authentication of the initiator to the server, as the server does not send any challenge to the initiator to respond. Fortunately, this does not compromise the security between the initiator and the responder.

Anonymity of the devices: In all the proposed protocols, the communicating devices always use the updated pseudonyms in the transmissions. This ensures the anonymity.

Unlinkability among devices' successful connections: In all the proposed protocols, the initiator D_i always generates its pseudonym as $PN = h_2(N_i)$ and updates its seed $N_i = h_1(N_i)$, where h_1 and h_2 are two independent one-way hashing functions. The preimage resistance property of the one-way hashes ensures the unlinkability of the initiator. Regarding the responder $D_{\bar{\imath}}$'s pseudonyms, the server finds $D_{\bar{\imath}}$'s current pseudonym $PN_{\bar{\imath},\bar{k}}$ in its local pseudonym vectors to connect $D_{\bar{\imath}}$, and then updates the seed and pseudonym for the next session; this $PN_{\bar{\imath},\bar{k}}$ may lag behind the active pseudonym at $D_{\bar{\imath}}'s$ side, due to possible attacks or unreliable transmissions. Fortunately the lag-behind of $PN_{\bar{\imath},\bar{k}}$ at the server's side also implies that this $PN_{\bar{\imath},\bar{k}}$ in the previous sessions does not result in any successful connection; therefore, each pseudonym could only result in at most one successful authentication. This ensures the property of "unlinkability among devices' successful connections".

Resistance to desynchronization-based DoS attacks: In our designs, each device (say D_i) keeps its active level-1/level-2 pseudonyms and the past $w1$ level-1 pseudonyms/the past $w2$ level-2 pseudonyms while the server or a responding device (say $D_{\bar{\imath}}$) keeping its recorded $D_{\bar{\imath}}'s$ active pseudonyms and the future $w1$ level-1 (and $w2$ level-2) pseudonyms. This arrangement facilitates the two parties to cope with the situations that $D_{\bar{\imath}}'s$ active pseudonym at the two communicating entities are not synchronized, due to possible attacks or unreliable connections. The robustness of desynchronization-based-DoS-resistance depends on the window size ($w1$ and $w2$); the larger the size, the stronger robustness (also the larger storage space). The choice of the size should consider the threat severity, the desirable robustness, and the storage space.

Perfect forward secrecy of the session keys: Each session key calculation involves the secret key and a random-and-fresh Diffie–Hellman value xyP, which ensures the forward secrecy of the session keys.

5.2. The Performance Evaluation

In this section, we evaluated the performance of the proposed schemes, and compare them with its counterparts. Here, we concern the computation cost and the communication cost (the message lengths and the number of message rounds). Even though there are many works addressing either the D2S authentication or the D2D authentication, their target scenarios and functions are quite different. Therefore, we have to sort out the scenarios and the functions first, and then compare our schemes with the counterparts to have a fair comparison.

Table 3 lists the target scenarios and functions of several D2S/D2D authentication schemes. In Table 3, Column 2 lists the target scenarios. Chien [11] focuses on the D2S authentication only, and Chien [25] addresses both the D2S authentication and the D2S group authentication; our proposed schemes and the schemes [15,16,18] address the D2D authentication, while the schemes [13,20] addressing the D2D group authentication. Sec-D2D [18] proposed a near-field D2D authentication, using accelerator sensors, speakers, and microphones; it fits only the near-field D2D cases where the owner of the device should actively participate the authentication process; it is a special application case, and, therefore, we would not include it in the detailed performance comparison. The preproof version [21] still has several typos and errors in the equations, and it would not be included in the comparison.

Table 3. Brief summary of the targeted scenarios and the security goals of D2S/D2D authentication schemes.

	Target Scenario	Security Goals	Mechanism for Anonymity	Cryptographic Alg.
Ours	D2S authentication/D2D authentication	D2S authentication/D2D authentication/anonymity/unlinkability	2LCH anonymous identification	Hash/DH$_{ECC}$/Sym.
Chien [11,25]	D2S authentication[11,25]/D2S group authentication [25]	D2S authentication/D2S group authentication	Not consider	Hash/ECC/pairing [25]
Fang et al. [12] [1]	D2S authentication/D2S group authentication	D2S authentication/D2S group authentication/unlinkability of the level-1 device authentication/anonymity of level-2 devices	Level-1 IoT: pseudonym-bound CLPK Level-2 IoT: fixed pseudonyms	Level-1 IoT: Hash/Sym./ECC-CLPK. Level-2 IoT: Hash/Sym.
Wang-Yan [13]	D2D group authentication	D2D authentication/D2D group authentication/anonymity	Pseudonym-bound PKC	Hash/pairing-based PKC/Sym.
SeDS [15]	D2D authentication/data sharing	D2D authentication/non-repudiation/ anonymity	pseudonym-bound PKC	Hash/sym./pairing-PKC
AAKA-D2D [16]	D2D authentication	D2D authentication/anonymity	pseudonym-bound PKC	Hash/sym./pairing-PKC
Sec-D2D [18]	Near field D2D authentication	D2D authentication	Not consider	Hash/Physical Layer Security (acceleration sensor/speaker/microphone)
Shang et al. [20]	D2D group authentication	D2D group/anonymity	Pseudonym-bound CLPK	Hash/Sym./ECC-CLPK
Sun et al. [22]	D2S authentication/D2D group authentication	D2S authentication/D2D group authentication/anonymity/unlinkability	Pairing-based IDPE [26]	Hash/Sym./signature/batch signature/pairing-based IDPE

Authentication: authentication; alg.: algorithm; PKC: Public-Key Cryptography; ECC: Elliptic Curve Cryptography; DH$_{ECC}$: Diffie–Hellman key using ECC; Sym.: symmetric key cryptography; pairing-PKC: PKC using pairing operations; pseudonym-bound PKC: PKC bound to entities' pseudonym; ECC-CLPK: Certificate Less Public Key system using ECC; pairing-based IDPE: identity-based prefix encryption using pairing. [1] In the D2S group authentication [12], one compromised level-2 device can impersonate any other devices in the group as the session key computation only use public values and one single group common secret.

In Table 3, Column 3 lists the main security goals, and Column 4 specifies the mechanisms to achieve the anonymity if it applies. Our schemes and the schemes

[12,13,15,16,18,22] all address the anonymity challenge while only our schemes and [22] tackling the unlinkability challenge. To achieve anonymity, the schemes [12,13,15,16,18,22] adopt pseudonyms, and bind either PKC certificates or certificateless public keys (CLPKs) [12,22] to the pseudonyms. These pseudonyms and the corresponding certificates (or CLPKs) are valid for a period of time, and need to be renewed periodically [13,15,16,18]. That is why they can only achieve anonymity but not unlinkability. This approach of using pseudonym-bound certificates or pseudonym-bound CLPKs have several critical weaknesses; first, correlating transmissions of the same pseudonyms can link the transmissions and might reveal sensitive information about the entities; second, it is costly to renew the certificates and CLPKs. In [12], the level-1 IoT devices are responsible for the registration and management of the level-2 IoT devices; therefore, the level-1 devices are more like a server but not resource-limited IoT devices; furthermore, a single compromised level-2 device can impersonate any other devices in the D2S group authentication, because the session key computations only use the public values and one single group secret. The D2D group authentication of [22] use lots of intensive computations like signatures, batch signatures, and especially the pairing-based IDPE [26]. Column 5 lists the cryptographic algorithms used. We can see that several of them adopt pairing-based PKCs or pairing-based CLPKs; these schemes demand intensive computations.

After identifying the target scenarios and the security goals in Table 3, we now compare the performances of our schemes with their counterparts in Table 4. Let T_h denote the computation cost for one hashing; T_{Ecc_M} denotes the cost for one point multiplication in the ECC setting; T_{Ecc_A} denotes the cost for one point addition in the ECC setting; T_{exp} denotes the cost for one exponentiation in the GF(p) field; T_{mlp} denotes the cost for one multiplication in the GF(p) field; T_{ENC} denotes that for either one symmetric encryption or one decryption; T_{pair} denotes that for one pairing computation; T_{SIG} denotes that for one ordinary signature generation or one signature verification; and T_{PE} denotes that for one identity-based prefix encryption. The second column of Table 4 summarizes the computational cost. In our schemes, even though the server or any devices need to initialize and update the pseudonym vectors (which require $2w1$ T_h initially), the amortized cost per session is only a few number of hashing. From Table 4, we can see that the level-2 IoT D2S authentications of [12] demand the least computational complexity; but we should note that the schemes use the fixed pseudonyms, and a single compromised device can impersonate any other group members in the D2S group authentication. Sun et al.'s direct D2D group authentication [22] demands lots of intensive computations like exponentiations, encryptions, digital signatures, and especially the pairing-based IDPE [26]. The number of pairing computations of the pair-based IDPE also depends on the number of the items in the identity string [26].

Regarding the communication cost, we concerned two metrics: the message bit length and the number of message rounds. Let L_{ID} and L_h respectively denote the bit length of one identity and one hash, and L_{SIG} denote the bit length of one ordinary digital signature. We let the bit length of one random challenge be the same as one hash. We also let the bit length of one encryption be the bit length of its plaintext; for example, the bit length of $Enc_{K_i}(D_{\bar{\imath}}, PN)$ be $1L_{ID} + 1L_h$. The message lengths of each scheme are summarized in Column 4 of Table 4. For example, the total bit length of our D2S anonymous authentication is $9L_h + 3L_{ID}$ under the parameter setting we specify, where MAC is implemented using the HMAC algorithm, and the hash function is assumed to be SHA-256. The bit length of the initial request from the device to the server in our D2S anonymous authentication is $3L_h + 1L_{ID}$. So, if we assume the bit length of an identity is 80, then the bit length of the initial request and the total bit length of our D2S anonymous authentication are respectively 848 and 2544.

In Table 4 we further differentiated D2D schemes into either D2S2D cases or direct D2D cases. Chein's schemes [11,25] adopt the concept of time zone in which the entities share the same authorized tokens to derive the successive session keys; therefore, the cost per session is amortized and is greatly reduced; for example, Chien's scheme [11] apply

pairing operation, but the cost is amortized among m times of accesses within a time zone; so the amortized cost of a device per session is $(6 + 2/m)T_h + 1T_{ECCM} + 1/m\ T_{pair}$; when m (the number of requests within one time zone) is large enough, the amortized pairing computation $1/m\ T_{pair}$ could be greatly reduced.

Table 4. Summary of the computational cost and the communication cost per session.

	Schemes	Computation	Communication
Ours	D2S authentication	D_i: $6 \times\ T_h + 2T_{ECCM}$ (0.597 ms); S: $6 \times\ T_h + 2T_{ECCM}$	$9L_h + 3L_{ID}$; 3 msg. steps
	S-assist-always D2S2D authentication	D_i: $7 \times\ T_h + 2T_{enc} + 2T_{ECCM}$ (0.597 ms); $D_{\bar{i}}$: $6 \times\ T_h + 1T_{enc} + 2T_{ECCM}$; S: $6 \times\ T_h + 3T_{enc}$	$18L_h + 7L_{ID}$; msg. steps
	S-assisted-once-a-while D2S2D authentication	D_i: $8 \times\ T_h + 2T_{enc} + 2T_{ECCM}$ (0.597 ms); $D_{\bar{i}}$: $6 \times\ T_h + T_{enc} + 2T_{ECCM}$ S: $9 \times\ T_h + 3T_{enc}$	$22L_h + 7L_{ID}$; 4 msg. steps
	Direct D2D	D_i: $7 \times\ T_h + 2T_{ECCM}$ ((0.597 ms)); $D_{\bar{i}}$: $7 \times\ T_h + 2T_{ECCM}$	$9L_h + 7L_{ID}$; 3 msg. steps
Fang [12]	Level-1 IoT: D2S authentication	D_i: $5 \times\ T_h + 6T_{ECCM}$ (1.79 ms)	$12L_h$; 3 msg. steps
	Level-2 IoT: D2S authentication [3]	D_i: $2 \times\ T_h + 1T_{ENC}$ (0.00008 ms)	$2L_h + 2L_{ID} + 1L_{ENC}$; 3 msg. steps
	Level-2 IoT: D2S group authentication [4]	D_i: $3 \times\ T_h + 2T_{ENC}$ (0.0014 ms)	$(2n + 1)L_h + (2n + 1)L_{ID} + 2nL_{ENC}$, where n is the number of the group; 7 msg. steps
SeDS [15] [2]	D2S2D authentication + data sharing	D_i (transmitter): $2 \times\ T_h + 5T_{exp} + 2T_{ENC}$ (1.709 ms); $D_{\bar{i}}(requestor)$: $4 \times\ T_h + 1T_{exp} + 4T_{pair}$ (55.036 ms)	$16L_h + 21L_{ID} + (2L_{ENC} + 2L_{sig})$; 8 msg. steps
AAKA-D2D [16]	D2S2D authentication	D_i:$3 \times\ T_h + 5T_{exp} + 3T_{ENC} + 2T_{pair}$ (29.056 ms);	$10L_h + 12L_{ID}$; 7 msg. steps
Sun et al. [22]	Direct D2D group authentication	$D_i(initiator)$: $(n + 3) \times\ T_h + nT_{exp} + nT_{mlp} + (2n + 1)T_{ECCM} + (2n + 2)T_{ECCA} + T_{SIG} + T_{PE} + nT_{ENC}$	$(n + 3)L_h + (5n + 5)L_{ID} + nL_{ENC} + 1L_{SIG}$; $(n + 2)$ msg. steps
Chien [25]	D2S authentication	$(6+2/m)\ T_h + 2/m\ T_{ECCM}$ [1] (0.64 ms for $m = 40$; 0.31 ms for $m = 1200$)	$4L_h + 1L_{ID} + (4L_h + 3L_{ID})/m$; 3 msg. steps

[1] m denotes the number of services one device access within the specified time-zone; for example, {$m = 40$ with time zone = 1 day} means "a device issues 40 requests per day", which represents those cases of which a device seldom issues requests; {$m = 1200$ with time zone = 1 day } means "a device issues 1200 requests per day", which represents those cases of which a device issues near one request per minute. [2] SeDS scheme [15] include entity authentication and data sharing; therefore, the operations and the message overhead are much larger. [3] In the level-2 IoT to the level-1 IoT authentication of [12], the level-1 device acts as a server. [4] In the group of level-2 IoTs to the service provider authentication, the level-1 device acts as a relay.

To our knowledge, there is no one single experiment covering all the execution times of all the operations our schemes use. Therefore, to further compare the computation cost, we refer to the figures from the two implementations [24,44], where the bit length of Galois field is 1024 bits, the ECC group G is of order $|q| = 160$, the hash is SHA-256, and the symmetric encryption is AES-CCM. In Table 2 of [24], they list the execution time of the pairing, the exponentiation, and the scalar multiplication for such a setting; in Table 5 of [44], they list the execution cycles of several operations including the encryption, the hashing, and the scalar multiplication. Combining the two tables, we can estimate the required execution times of the operations for comparison. Table 5 lists the computational costs (in

ms) for such a setting. The 3rd column of Table 4 lists some estimated cost (in ms) when we apply the figures from Table 5; for example, our D2S authentication demands D_i $6T_h + 2T_{ECCM}$ (0.597 ms), where 0.597 ms represents the estimated time when we apply the figures from Table 5 into $6T_h + 2T_{ECCM}$; AAKA-D2D's D2S2D authentication demands 29.056 ms.

Table 5. Time cost (in ms) for the operations.

T_{pair}	T_{exp}	T_{ECC_M}	T_{ENC}	T_h
13.6736	0.3418	0.2986	0.00004	0.00002

Figure 13 shows the bar charts of the estimated computation time (in ms), using the figures from Table 5. The left four items are the proposed schemes. From Figure 13, we can see that the computation times of our schemes, Chien's schemes [25], and Fang et al.'s level-2 D2S group authentication are almost negligible, compared to those of Se-DS [15] and AAKA-D2D [16]. We can see that those pairing-based schemes [15,16,22] demand much larger computational costs. Of course, we have to note that the cost of Se-DS [15] include the pairing-based signatures and verifications operations for additional data sharing.

In a short summary of the performance comparison both our proposed schemes, Chien's schemes [11,25], and Fang et al.'s level-2 D2S authentication [12] demand almost negligible computational cost, compared to other PKC-based schemes. However, Chien's schemes [11,25] do not provide anonymity, Fang et al.'s level-2 D2S authentication [12] does not consider unlinkability, and the D2S group authentication [12] is vulnerable to the single-compromised-device-impersonation attack. Those pseudonym-bound-certificate schemes [29,31,32] and the pseudonym-bound-CLPK schemes [36] demand much larger computational overhead, which make them much less attractive to the resource-limited devices; additionally, they only provide anonymity but not unlinkability; the process of periodically renewing the certificates or the CLPKs is also very costly.

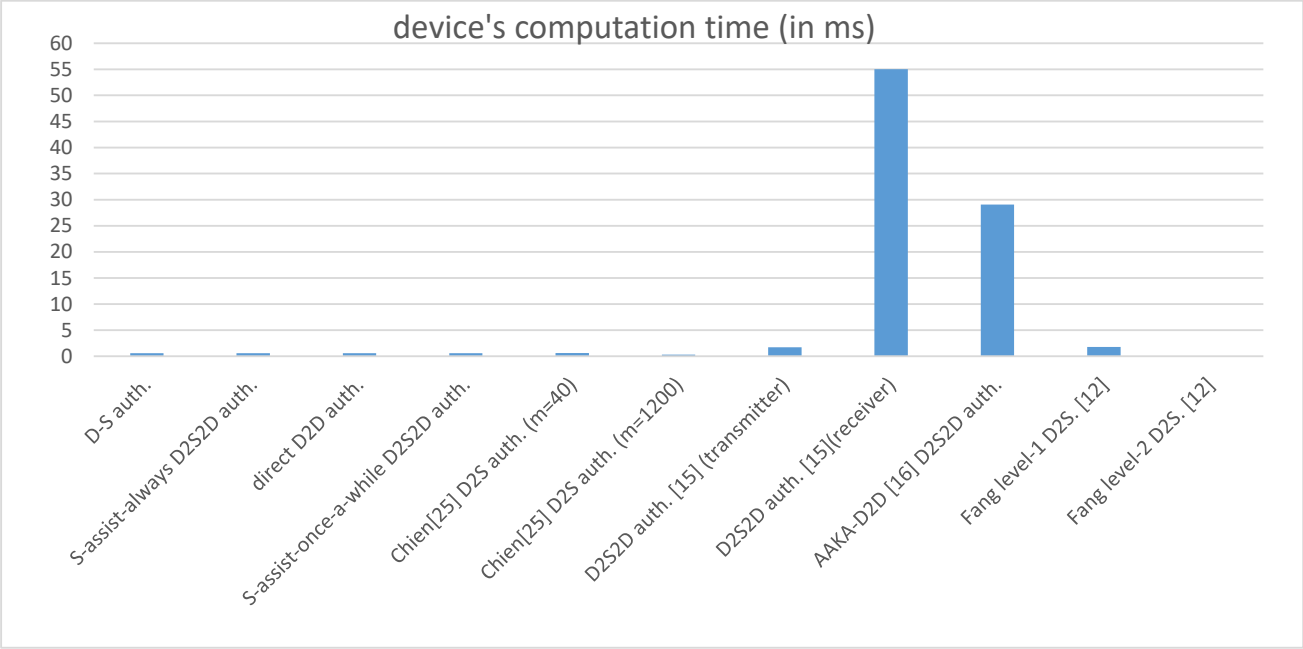


Figure 13. The estimated computation time of a device using the figures from Table 5.

6. The Conclusions

In this paper, we proposed the 2-Level-Composite-Hash (2LCH) to facilitate anonymous identification. Based on the 2LCH, we proposed the anonymous device authentication schemes for the device-to-server (D2S) scenarios, for the device-to-server-to-device (D2S2D) scenarios and for the direct device-to-device (D2D) scenarios. We showed the application of our schemes on the MQTT context. The design of the 2LCH effectively and efficiently facilitates anonymous device authentication using only simple hashing; it can effectively tackle both the desynchronization-based DoS attacks and the unreliable connection issues. The evaluations show that all our proposed schemes demand almost negligible computational costs, compared to the conventional pseudonym-bound PKC and pseudonym-bound CLPK schemes. Our schemes do not require the costly on-line certificate-renewing process. All these excellent merits make the proposed schemes very suitable for those resource-constrained IoT devices.

There are several interesting challenges for the future work. How do we optimize the window size of the pseudonym vector under different scenarios? Another is how could we improve the storage efficiency of the level-2 pseudonym vectors of a device, especially when a device simultaneously builds several direct D2D sessions with several devices. The other is to extend the current works to the cases where the devices belong to different service domains.

Funding: This project is partially supported by the Ministry of Science and Technology of Taiwan, under the grant No. MOST 108-2221-E-260-009-MY3.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Donno, M.D.; Dragoni, N.; Giaretta, A.; Spognardi, A. DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation. *Secur. Commun. Netw.* **2018**, doi:10.1155/2018/7178164.
2. Yang, Y.; Wu, L.; Yin, G.; Li, L.; Zhao, H. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet Things J.* **2017**, *4*, 1250–1258.
3. Avispa—A tool for Automated Validation of Internet Security Protocols. Available online: http://www.avispa-project.org (accessed on 20 February 2020).
4. OASIS Message Queuing Telemetry Transport (MQTT). Available online: https://www.oasis-open.org/committees/mqtt/ (accessed on 7 July 2020).
5. Zhang, R. Enhanced authentication—MQTT 5.0 new features EMQ. Available online: https://www.emqx.io/blog/mqtt5-enhanced-authentication (accessed on 7 December 2020).
6. Andrea, I.; Chrysostomou, C.; Hadjichristofi, G. Internet of Things:Security vulnerabilities and challenges. In Proceedings of the IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015; pp. 180–187.
7. Lin, C.I.; Uusitalo, M.A.; Moessner, K. The 5G Huddle: From the guest editors. *IEEE Veh. Technol. Mag.* **2015**, *10*, 28–31.
8. Soldani, D.; Manzalini, A. Horizon 2020 and Beyond: On the 5G Operating System for a True Digital Society. *IEEE Veh. Technol. Mag.* **2015**, *10*, 32–42.
9. Li, J.; Wen, M.; Zhang, T. Group-Based Authentication and Key Agreement with Dynamic Policy Updating for MTC in LTE-A Networks. *IEEE Internet Things J.* **2016**, *3*, 408–417.
10. Lai, C.Z.; Lu, R.-X.; Zheng, D.; Li, H.; Shen, X.M. Toward Secure Large-scale Machine-to-machine Communications in 3GPP Networks: Challenges and Solutions. *IEEE Commun. Mag.* **2015**, *53*, 12–19.
11. Chien, H.Y. An Effective Approach to Solving Large Communication Overhead Issue and Strengthening the Securities of AKA Protocols. *Int. J. Commun. Syst.* **2017**, doi:10.1002/dac.3381.
12. Fang, D.; Qian, Y.; Hu, R.Q. A Flexible and Efficient Authentication and Secure Data Transmission Scheme for IoT Applications. *Ieee Internet Things J.* **2020**, *7*, 3474–3484.
13. Wang, M.; Yan, Z. Privacy-Preserving Authentication and Key Agreement Protocols for D2D Group Communications. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3637–3647.
14. Islam, M.M.; Zhang, Z.S. Security Enhancement of D2D Communication Based on Handshaking Mechanism. In Proceedings of the 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 12–15 June 2019; pp. 484–489.
15. Zhang, A.; Chen, J.; Hu, R.Q.; Qian, Y. SeDS: Secure Data Sharing Strategy for D2D Communication in LTE-Advanced Networks. *IEEE Trans. Veh. Technol.* **2016**, *65*, 2659–2672.

16. Wang, M.; Yan, Z.; Song, B.; Atiquzzaman, M. AAKA-D2D: Anonymous Authentication and Key Agreement Protocol in D2D Communications. In Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/ UIC/ATC/CBDCom/IOP/SCI), Leicester, UK, 19–23 August 2019; pp. 1356–1362.
17. 3GPP System Architecture Evolution (SAE); Security architecture. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2296 (accessed on 14 March 2010).
18. Cao, M.; Wang, L.; Xu, H.; Chen, D.; Lou, C.; Zhang, N.; Zhu, Y.; Qin, Z. Sec-D2D: A Secure and Lightweight D2D Communication System With Multiple Sensors. *IEEE Access* **2019**, *7*, 33759–33770.
19. Van Oorschot, P.C. An alternate explanation of two BAN-logic 'failures'. In Proceedings of the Eurocrypt'93, Lofthus, Norway, 23–27 May 1993; Volume 765, pp. 443–447.
20. Shang, Z.; Ma, M.; Li, X. A Certificateless Authentication Protocol for D2D Group Communications in 5G Cellular Networks. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–7.
21. Mahapatra, B.; Turuk, A.K.; Nayyar, A. Multilevel authentication and key agreement protocol for D2D communication in LTE based C-IoT network. *Microprocess. Microsyst.* **2021,** pre-proof version, doi:10.1016/j.micpro.2020.103720.
22. Sun, Y.; Cao, J.; Ma, M.; Zhang, Y.; Li, H.; Niu, B. EAP-DDBA: Efficient Anonymity Proximity Device Discovery and Batch Authentication Mechanism for Massive D2D Communication Devices in 3GPP 5G HetNet. *IEEE Trans. Dependable Secur. Comput.* **2020**, doi:10.1109/TDSC.2020.2989784.
23. Morabito, R.; Petrolo, R.; Loscri, V.; Mitton, N. LEGIoT: A Lightweight Edge Gateway for the Internet of Things. *Future Gener. Comput. Syst.* **2018**, *8*, 1–15.
24. Li, X.; Liu, S.; Wu, F.; Kumari, S.; Rodrigues, J.J.P.C. Privacy Preserving Data Aggregation Scheme for Mobile Edge Computing Assisted IoT Applications. *IEEE Internet Things J.* **2019**, *6*, 4755–4763.
25. Chien, H.Y. Group-Oriented Range-Bound Key Agreement for Internet-of-Things Scenarios. *IEEE Internet Things J.* **2018**, *5*, 1890–1903.
26. Lewko, A.; Waters, B. Why Proving HIBE Systems Secure Is Difficult. In Proceedings of the EUROCRYPT'14, Copenhagen, Denmark, 11–15 May 2014; pp. 58–76.
27. Chaum, D.; Heijst, E. "*Group signature", Advance in Cryptography-EUROCRYPT'91*; LNCS547; Springer: Berlin/Heidelberg, Germany, 1992; pp. 257–265.
28. Rivest, R.L.; Shamir, A.; Tauman, Y. How to Leak a Secret. In *Proceedings of the Asiacrypt'01*; LNCS 2248; Springer: Berlin/Heidelberg, Germany, 2001; pp. 552–565.
29. Bellare, M.; Sesai, A.; Pointcheval, D.; Rogaway, P. Relations among notations of security for public key Encryption schemes. In *Proceedings of the CRYPTO'98*; LNCS 1462; Springer: Berlin/Heidelberg, Germany, 1998; pp. 26–45.
30. Ohkubo, M.; Suzki, K.; Kinoshita, S. Cryptographic Approach to 'Privacy-Friendly' Tags. In RFID Privacy Workshop, 2003. http://www.avoine.net/rfid/download/slides/OhkuboSK-2003-mit-slides.pdf (accessed on 7 June 2020).
31. Safavi-Naini, R.S.; Seberry, J.R. Error-correcting codes for authentication and subliminal channels. *IEEE Trans. Inf. Theory* **1991**, *37*, 13–17.
32. Park, C.S. Authentication protocol providing user anonymity and untraceability in wireless mobile communication systems. *Comput. Netw.* **2004**, *44*, 267–273.
33. Chien, H.Y.; Laih, C.S. ECC-Based Lightweight Authentication Protocol with Un-traceability for Low-Cost RFIDs. *J. Parallel Distrib. Comput.* **2009**, *69*, 848–853.
34. Chien, H.Y. Combining Rabin Cryptosystem and Error Correction Codes to Facilitate Anonymous Authentication with Untraceability for Low-End Devices. *Comput. Netw.* **2013**, *57*, 2705–2717.
35. Chien, H.Y. Highly Efficient Anonymous IoT Authentication Using Composite Hashing. In Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing, Aizuwakamatsu, Fukushima, Japan, 30 January–2 February 2021.
36. Alezabi, K.A.; Hashim, F.; Hashim, S.J.; Ali, B.M. An efficient authentication and key agreement protocol for 4G (LTE) networks. In Proceedings of the 2014 IEEE REGION 10 SYMPOSIUM, Kuala Lumpur, Malaysia, 14–16 April 2014; pp. 502–507.
37. Burrows, M.; Abadi, M.; Needham, R. A logic of authentication. In Proceedings of the 12th ACM symposium on Operating systems principles, Litchfield Park, AZ, USA, 3–6 December 1989; pp.1–13.
38. Basin, D.; M¨odersheim, S.; Vigan`o, L. OFMC: A Symbolic Model-Checker for Security Protocols. *Int. J. Inf. Secur.* **2005**, *4*, 181–208.
39. Von Oheimb, D. Specification language HLPSL developed in the EU project AVISPA. In Proceedings of the APPSEM 2005 Workshop, Frauenchiemsee, Germany, 12–15 September 2005.
40. Lesjak, C.; Hein, D.; Hofmann, M.; Maritsch, M.; Aldrian, A.; Priller, P.; Ebner, T.; Ruprechter, T.; Pregartne, G. Securing Smart Maintenance Services: Hardware-Security and TLS for MQTT. In Proceedings of the 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, UK, 22–24 July 2015, doi:10.1109/INDIN.2015.7281913.
41. Chien, H.Y.; Chen, Y.J.; Qiu, G.H.; Liao, J.F.; Hung, R.W.; Kuo, X.A.; Lin, P.C.; Chiang, M.L.; Su, C.H. A MQTT-API-Compatible IoT Security-Enhanced Platform. *Int. J. Sens. Netw.* **2020**, *32*, 54–68.

42. Bellare, M.; Canetti, R.; Krawczyk, H. Keying Hash Functions for Message Authentication. In *Advances in Cryptology—CRYPTO '96*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 1–15.
43. ISO/IEC 20922:2016. Information technology—Message Queuing Telemetry Transport (MQTT) v3.1.1. Available online: https://www.iso.org/standard/69466.html (accessed on 7 June 2020).
44. De la Piedra, A.; Braeken, A.; Touhafi, A. A performance comparison study of ECC and AES in commercial and research sensor nodes. In Proceedings of the Eurocon 2013, Zagreb, Croatia, 1–4 July 2013; pp. 347–354.