Access Control in Internet of Things (IoT) Applications

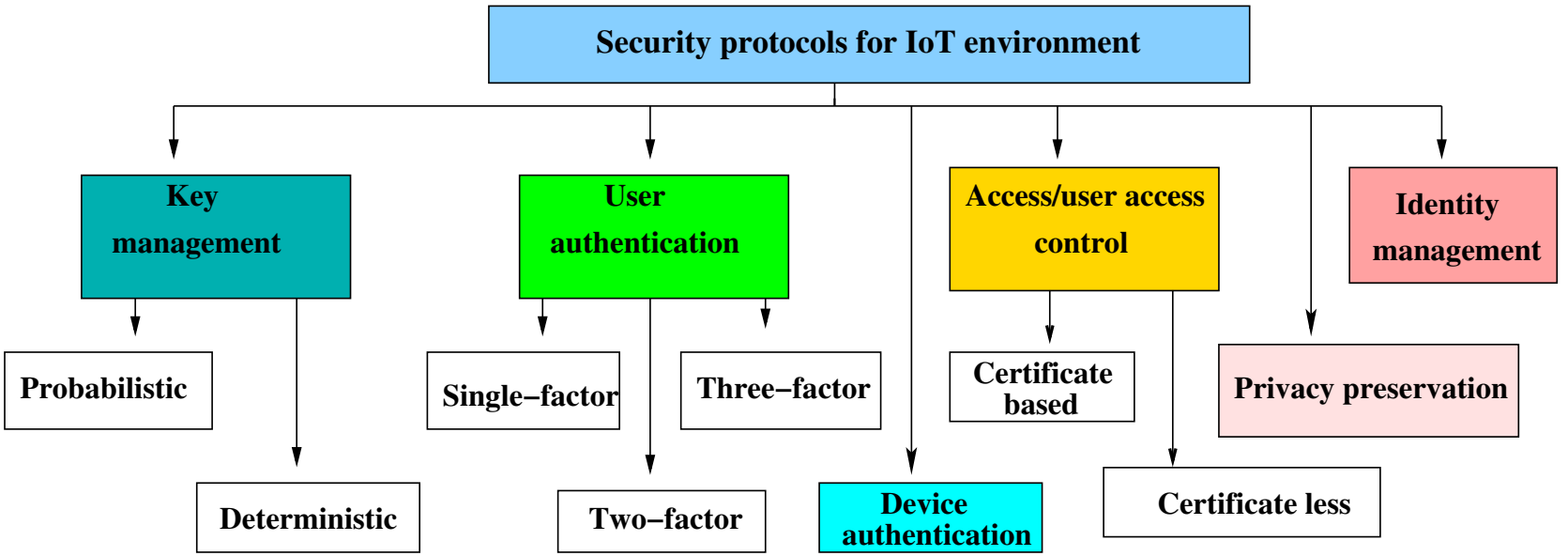
Dr. Ashok Kumar Das

IEEE Senior Member
Associate Professor
Center for Security, Theory and Algorithmic Research
(Department of Computer Science and Engineering)
International Institute of Information Technology, Hyderabad
(Formerly **Indian Institute of Information Technology, Hyderabad**)

E-mail: *ashok.das@iiit.ac.in*
Homepage: http://www.iiit.ac.in/people/faculty/ashokkdas
Personal Homepage: https://sites.google.com/view/iitkgpakdas/

September 25, 2022

Taxonomy of security protocols in IoT



Ashok Kumar Das, Sherali Zeadally, and Debiao He. "Taxonomy and Analysis of Security Protocols for Internet of Things," in ***Future Generation Computer Systems (Elsevier)***, Vol. 89, pp. 110-125, 2018, DOI: 10.1016/j.future.2018.06.027. (2021 SCI Impact Factor: 7.307)

Access Control in IoT

- A deployed IoT smart device may not always be legitimate because some illegal (malicious) IoT devices can be placed by an adversary in IoT environment.
- In this case, it turns out to be hard task in distinguishing illegal new smart devices from the legal smart devices in IoT environment.
- Hence, a "device access control mechanism" is very much required when new smart devices are installed/deployed in IoT environment for preventing illegal devices from entering the network.
- For this issue, we consider the device access control in IoT environment with the following two tasks:
  - ***Device (node) authentication:*** This task needs that "a newly deployed smart device must authenticate itself to its neighbor smart devices to prove that it is a legitimate smart device for accessing the information from the other smart devices".
  - ***Key management:*** This task needs that "a newly deployed smart device needs to establish shared secret keys with its existing neighbor smart devices to assure secure communication during the transmission of sensing information".

Certificate-Based Device Access Control for IoT Environment

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

- **Saurav Malani, Jangirala Srinivas, Ashok Kumar Das, Kannan Srinathan, and Minho Jo.** "Certificate-Based Anonymous Device Access Control Scheme for IoT Environment," in ***IEEE Internet of Things Journal***, Vol. 6, No. 6, pp. 9762-9773, Dec. 2019, DOI: 10.1109/JIOT.2019.2931372. (2021 SCI Impact Factor: 10.238)

Network Model

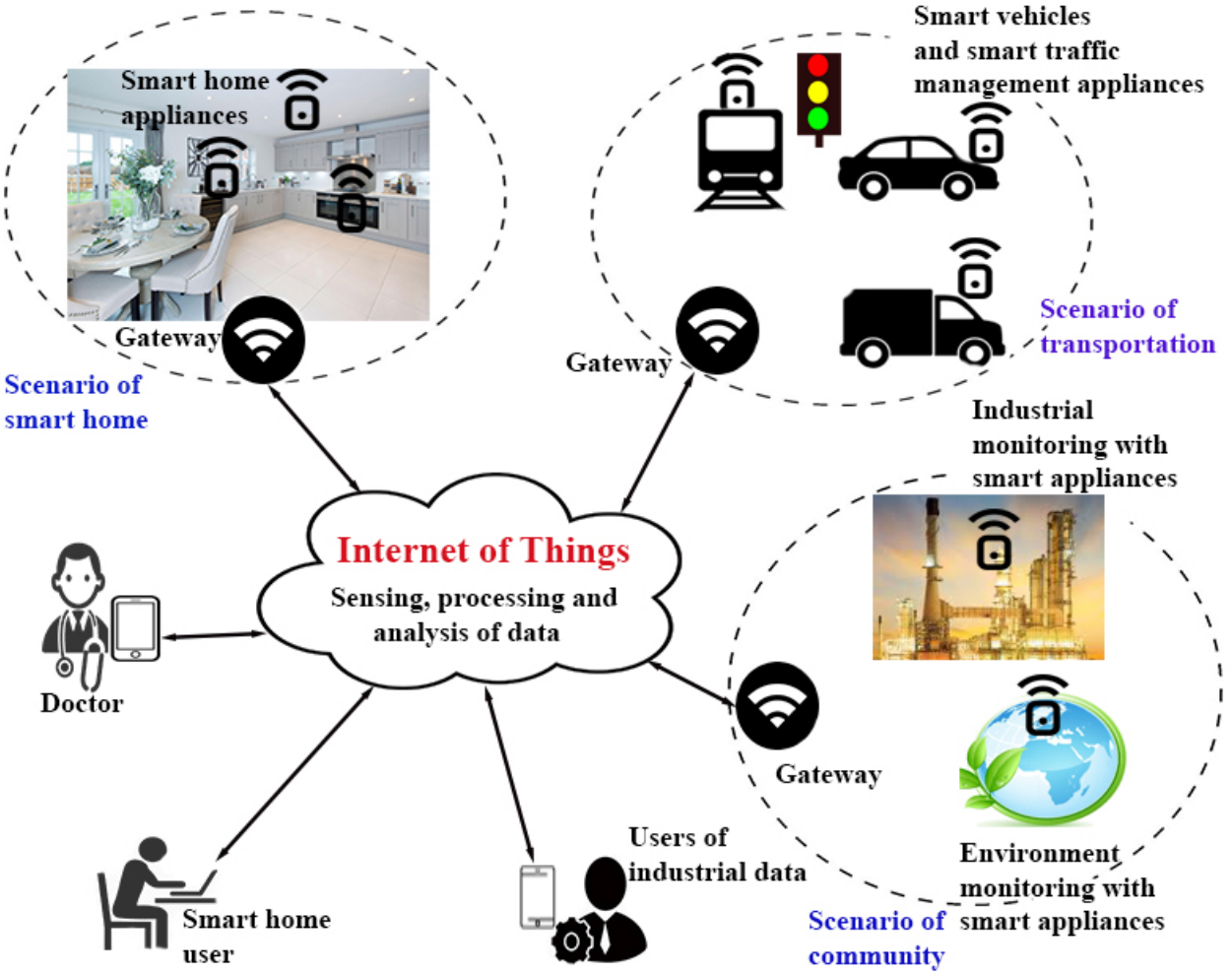


Figure: A generic IoT environment

Threat Model

- The “Dolev-Yao threat (DY) model” in one of the broadly-accepted models that can be applied in various networks, such as wireless sensor networks, general wire/wireless networks as well as IoT environment. The DY model assumes that the end-point communicating participants (e.g., IoT smart devices in our proposed scheme) can not be taken as trusted nodes in the network, and communication among the devices are insecure as their communication relies on wireless medium.
- Under the DY model, an adversary can also tamper the data being communicated among the participants including interception, modification, deletion and insertion of fake information in the communicated messages.
- Due to “hostile (unattended) environment” for some IoT applications, there is also a possibility of “physical smart devices capture attack” by the adversary by extracting the credentials stored in those captured devices by utilizing the power analysis attacks

Threat Model

- The widely-accepted “Canetti and Krawczyk’s adversary model (CK-adversary model)” becomes the “current *de facto* standard model in modeling authenticated key-exchange protocols”.
- According to the CK-adversary model, “the adversary can not only deliver the messages (as in the DY model), but also can compromise the secret credentials, secret keys and session states where these are stored in insecure memory”.
- Therefore, it becomes an essential requirement that “the leakage of some forms of secret credentials, such as session ephemeral secrets or secret key, should have the minimum possible consequence on the secrecy of other secret credentials of the communicating participants”.
- Finally, it is also a typical assumption that the *GWN* is trusted node and it will not be compromised by the adversary. Hence, the *GWN* can be put under a physical locking system in the IoT environment based on applications (e.g., smart home, healthcare and industrial IoT).

Notations and their significance

Symbol	Significance
SD_i, ID_{dev_i}	i^{th} IoT sensing device and its identity, respectively
GWN, ID_{gwn}	Gateway node and and its identity, respectively
$E_p(a, b)$	Non-singular elliptic curve: $y^2 = x^3 + ax + b \pmod p$
G	A base point in $E_p(a, b)$
$x.P$	"ECC scalar (point) multiplication" of the point $P \in E_p(a, b)$, $x.P = P + P + \cdots + P$ (x times), $x \in Z_p^*$
$P + Q$	"ECC point addition", $P, Q \in E_p(a, b)$
(p_{gwn}, P_{gwn})	Private and public keys pair of the GWN, respectively, $P_{gwn} = p_{gwn}.G$
(k_{dev_i}, K_{dev_i})	Private and public keys pair of SD_i, respectively, $K_{dev_i} = k_{dev_i}.G$
$cert_{dev_i}$	Certificate of SD_i generated by the GWN
$sign_{dev_i}$	Signature of SD_i signed by its private key k_{dev_i}
TS_i	Current timestamp of SD_i
ΔT	"Maximum allowable transmission delay associated with a message"
$SD_i \rightarrow SD_j$: M	Device SD_i sends message M to its neighbor device SD_j
$h(\cdot)$	"Collision-resistant one-way cryptographic hash function"
$\oplus$, $\|$	"Bitwise XOR" & "string concatenation" operations, respectively
$\mathcal{A}$	Passive/active adversary

Setup Phase

The setup phase is executed by the "gateway node (*GWN*)" in order to select the system parameters with the following steps. It is worth noting that the *GWN* can be considered as a "Public Key Infrastructure (PKI)" in the proposed scheme (DACS-IoT).

S1. The *GWN* fixes a "collision-resistant one-way cryptographic hash function" $h(\cdot)$ defined by h: $\{0,1\}^* \rightarrow \{0,1\}^l$, which takes an arbitrary length input string and produces a fixed-length (l bits) hash output, called the message digest or hash value.

S2. Next, the *GWN* picks a "non-singular elliptic curve $E_p(a,b)$ of the form $y^2 = x^3 + ax + b \pmod p$ having a large prime p and two constants $a, b \in Z_p = \{0, 1, \ldots, p-1\}$ such that the necessary and sufficient condition $4a^3 + 27b^2 \neq 0 \pmod p$ is satisfied".

S3. The *GWN* then picks a base point G on $E_p(a,b)$ whose order is as large as p, say n such that $n.G = G + G + \cdots + G$ (n *times*) $= \mathcal{O}$, where $\mathcal{O}$ is called the point at infinity or zero point.

S4. Finally, the *GWN* generates a public-private key pair (P_{gwn}, p_{gwn}) by selecting a private key p_{gwn} randomly & then calculating its correspondingly public key as $P_{gwn} = p_{gwn}.G$.

At the end of this phase, the *GWN* publishes the public parameters $\{h(\cdot), E_p(a,b), G, P_{gwn}\}$ and keeps its private key p_{gwn} as secret.

Device Enrollment Phase

In this phase, all the IoT sensing devices need to be registered in offline mode by the *GWN* before these are installed in the IoT environment. The following steps are essential to serve this purpose:

E1. For each IoT sensing device SD_i, the *GWN* fixes a unique ID ID_{dev_i} & generates a random private key $k_{dev_i} \in Z_p^*$ to calculate the corresponding public key $K_{dev_i} = k_{dev_i}.G$, where $Z_p^* = \{a \mid 0 < a < p, \gcd(a, p) = 1\} = \{1, 2, \ldots, p-1\}$. In addition, for SD_i, the *GWN* generates a cerificate key $ck_{dev_i} \in Z_p^*$ to calculate the corresponding public key $CK_{dev_i} = ck_{dev_i}.G$, and publishes CK_{dev_i}.

E2. The *GWN* then creates a certificate $cert_{dev_i}$ for every sensing device SD_i as $cert_{dev_i} = p_{gwn}.h(ID_{gwn} || K_{dev_i}) + ck_{dev_i} \pmod p$ using its own private key p_{gwn}. It is worth noting that since the certificate $cert_{dev_i}$ contains p_{gwn}, it is only created by the *GWN*.

E3. Finally, the *GWN* pre-loads the materials $\{ID_{dev_i}, ID_{gwn}, k_{dev_i}, K_{dev_i}, h(\cdot), E_p(a, b), G, cert_{dev_i}, P_{gwn}\}$ in the memory of each SD_i.

Device Enrollment Phase

IoT sensing device (SD_i)
$\{ID_{dev_i}, ID_{gwn}, k_{dev_i}, K_{dev_i}, h(\cdot), E_p(a,b), G, cert_{dev_i}, P_{gwn}\}$

Figure: Credentials stored in SD_i during device enrollment phase

Device Access Control Phase

For secure communication between two neighbor IoT sensing devices SD_i and SD_j, node authentication needs to be successful before establishment of the secret pairwise key between them. This is achieved by executing the following steps:

A1. $SD_i \rightarrow SD_j$: $msg_1 = \{cert_{dev_i}, sig_{dev_i}, TS_i, R_i, K_{dev_i}\}$

- A1.1. SD_i first generates the current timestamp TS_i and a random secret $r_i \in Z_p^*$ to calculate the corresponding public $R_i = h(r_i || ID_{dev_i} || k_{dev_i} || TS_i).G$ using its own stored private key k_{dev_i} and identity ID_{dev_i}.
- A1.2. SD_i calculates the signature sig_{dev_i} on r_i and ID_{dev_i} as $sig_{dev_i} = h(r_i || ID_{dev_i} || k_{dev_i} || TS_i) + k_{dev_i}. h(cert_{dev_i} || R_i || K_{dev_i} || ID_{gwn} || TS_i) \pmod p$.
- A1.3. SD_i then dispatches the message authentication request message $msg_1 = \{cert_{dev_i}, sig_{dev_i}, TS_i, R_i, K_{dev_i}\}$ to its neighbor SD_j over open channel.

Device Access Control Phase

A2. $SD_j \rightarrow SD_i$: $msg_2 = \{cert_{dev_j}, sig_{dev_j}, TS_j, R_j, K_{dev_j}, SKV_{ij}\}$
After receiving the msg_1 from SD_i, SD_j performs the following sequence of verification steps:

A2.1. SD_j firstly checks if the verification condition $|TS_i - TS_{current}| < \Delta T$ holds or not, where $TS_{current}$ is the time when the message was received by SD_j and ΔT is the "maximum allowable transmission delay". If this verification does not hold, SD_j discards the message and discontinues the phase promptly. Once this condition is satisfied, the certificate $cert_{dev_i}$ verification is done by checking the condition: $cert_{dev_i}.G = h(ID_{gwn}||\ K_{dev_i})P_{gwn} + CK_{dev_i}$. Note that

$$\begin{aligned} cert_{dev_i}.G &= h(ID_{gwn}||K_{dev_i})(p_{gwn}.G) + ck_{dev_i}.G \\ &= h(ID_{gwn}||K_{dev_i}).P_{gwn} + CK_{dev_i}. \end{aligned}$$

After the successful certificate verification, the signature verification is done by SD_j to check if the condition $sig_{dev_i}.G = R_i + h(cert_{dev_i}||\ R_i||\ K_{dev_i}||\ ID_{gwn}||\ TS_i).K_{dev_i}$ holds true. If the condition fails, the phase is discontinued promptly. If the signature validation fails, the phase is also discontinued promptly.

Device Access Control Phase

A2.2.

- SD_j now generates a random secret $r_j \in Z_p^*$ and the current timestamp TS_j, and calculates $R_j = h(r_j || ID_{dev_j} || k_{dev_j} || TS_j).G$.
- After that SD_j calculates the signature $sig_{dev_j} = h(r_j || ID_{dev_j} || k_{dev_j} || TS_j) + k_{dev_j}.h(cert_{dev_j} || R_j || K_{dev_j} || ID_{gwn} || TS_j) \pmod p$.
- Using the generated r_j, $cert_{dev_j}$ and TS_j, SD_j calculates the secret key shared with SD_i as $SK_{ij} = h(h(r_j || ID_{dev_j} || k_{dev_j} || TS_j).R_i || cert_{dev_i} || cert_{dev_j} || ID_{gwn})$ and its verifier as $SKV_{ij} = h(SK_{ij} || TS_j)$.
- Next, SD_j dispatches the authentication reply message $msg_2 = \{cert_{dev_j}, sig_{dev_j}, TS_j, R_j, K_{dev_j}, SKV_{ij}\}$ to SD_i over an open channel.

Device Access Control Phase

A3. After receiving the msg_2 from SD_j, SD_i performs the following sequence of verification stages to check the authenticity of the message msg_2 as well as SD_j.

A3.1. SD_i verifies the validity of $|TS_j - TS_{current}| < \Delta T$, where $TS_{current}$ represents the time when the message was received. Once this condition is fulfilled, the certificate $cert_{dev_j}$ verification is the next step by SD_i by checking the condition $cert_{dev_j}.G = P_{gwn}.h(ID_{gwn} || K_{dev_j}) + CK_{dev_j}$. After this certificate verification, the signature verification takes place using the condition $sig_{dev_j}.G = R_j + h(cert_{dev_j} || R_j || K_{dev_j} || ID_{gwn} || TS_j).K_{dev_j}$. If the condition fails, SD_i discontinues this phase promptly. Otherwise, the next step is executed by SD_i.

A3.2. SD_i calculates the secret key shared with SD_j as $SK'_{ij} = h(h(r_i || ID_{dev_i} || k_{dev_i} || TS_i).R_j || cert_{dev_i} || cert_{dev_j} || ID_{gwn})$ and the secret key verifier as $SKV'_{ij} = h(SK'_{ij} || TS_j)$, and checks if the computed secret key verifier SKV'_{ij} matches with the received verifier SKV_{ij}. If the condition fails, SD_j is not authenticated by SD_i. Otherwise, SD_i treats SK'_{ij} $(= SK_{ij})$ as the valid secret key shared with SD_j. In a similar way, SD_j also keeps the secret key SK_{ij} shared with SD_i.

Summary of Device Access Control Phase

Sensing Device (SD_i)	Sensing Device (SD_j)
Generate current timestamp TS_i. Randomly pick random secret r_i. Compute $R_i = h(r_i \|\| ID_{dev_i} \|\| k_{dev_i} \|\| TS_i).G$, $sig_{dev_i} = h(r_i \|\| ID_{dev_i} \|\| k_{dev_i} \|\| TS_i)$ $+k_{dev_i}.h(cert_{dev_i} \|\| R_i \|\| K_{dev_i} \|\|$ $ID_{gwn} \|\| TS_i) \pmod p$. $msg_1 = \{cert_{dev_i}, sig_{dev_i},$ $TS_i, R_i, K_{dev_i}\}$ $\longrightarrow$ (via open channel)	
	Check if $\|TS_i - TS_{current}\| \leq \Delta T$? If so, check if $cert_{dev_i}.G =$ $h(ID_{gwn} \|\| K_{dev_i}).P_{gwn} + CK_{dev_i}$ and $sig_{dev_i}.G = R_i + h(cert_{dev_i} \|\|$ $R_i \|\| K_{dev_i} \|\| ID_{gwn} \|\| TS_i).K_{dev_i}$? Pick random secret r_j & timestamp TS_j. Compute $R_j = h(r_j \|\| ID_{dev_j} \|\| k_{dev_j} \|\| TS_j).G$, signature $sig_{dev_j} = h(r_j \|\| ID_{dev_j}$ $\|\| k_{dev_j} \|\| TS_j) + k_{dev_j}.h(cert_{dev_j} \|\| R_j$ $\|\| K_{dev_j} \|\| ID_{gwn} \|\| TS_j) \pmod p$, secret key $SK_{ij} = h(h(r_j \|\| ID_{dev_j}$ $\|\| k_{dev_j} \|\| TS_j).R_i \|\| cert_{dev_i}$ $\|\| cert_{dev_j} \|\| ID_{gwn})$ and its verifier $SKV_{ij} = h(SK_{ij} \|\| TS_j)$. $msg_2 = \{cert_{dev_j}, sig_{dev_j}, TS_j,$ $R_j, K_{dev_j}, SKV_{ij}\}$ $\longleftarrow$ (via open channel)
Check if $\|TS_j - TS_{current}\| \leq \Delta T$? If so, verify if $cert_{dev_j}.G = P_{gwn}.h(ID_{gwn}$ $\|\| K_{dev_j}) + CK_{dev_j}$ & $sig_{dev_j}.G = R_j + h(cert_{dev_j}$ $\|\| R_j \|\| K_{dev_j} \|\| ID_{gwn} \|\| TS_j).K_{dev_j}$. If all conditions are satisfied, compute $SK'_{ij} = h(h(r_i \|\| ID_{dev_i}$ $\|\| k_{dev_i} \|\| TS_i).R_j \|\| cert_{dev_i}$ $\|\| cert_{dev_j} \|\| ID_{gwn})$ & verifier $SKV'_{ij} = h(SK'_{ij} \|\| TS_j)$. Check if $SKV'_{ij} = SKV_{ij}$? If so, SD_j is authenticated.	

Both SD_i and SD_j maintain secret key SK_{ij} $(= SK'_{ij})$.

Dynamic Device Addition Phase

To install a new IoT sensing device, say SD_{new} in the existing network, the *GWN* requires the following pre-deployment steps in the offline mode:

D1. The *GWN* needs to fix a unique ID, say ID_{new} & generate a random private key $k_{new} \in Z_p^*$ in order to calculate the corresponding public key $K_{new} = k_{new}.G$. In addition, for SD_{new}, the *GWN* generates a cerificate key $ck_{new} \in Z_p^*$ to calculate the corresponding public key $CK_{new} = ck_{new}.G$, and publishes CK_{new}.

D2. The *GWN* creates a certificate $cert_{new}$ for new device SD_{new} as $cert_{new} = p_{gwn}.h(ID_{gwn} || K_{new}) + ck_{new} \pmod{p}$ using its own private key p_{gwn}.

D3. The *GWN* then pre-loads the materials $\{ID_{new}, ID_{gwn}, k_{new}, K_{new}, h(\cdot), E_p(a, b), G, cert_{new}, P_{gwn}\}$ in the memory of SD_{new}.

Once the pre-deployment steps are performed successfully, the new sensing device SD_{new} can be deployed in the existing network and it starts secure communication with its neighboring sensing devices using the device access control phase.