Energy-Efficient Hybrid key Distribution Scheme for Wireless Sensor Networks

Majid Alshammari and Khaled Elleithy
Department of Computer Science and Engineering
University of Bridgeport, Bridgeport, CT

Abstract

Key distribution in Wireless Sensor Networks (WSNs) is challenging issue because a WSN is a network of resource-constrained nodes that carry limited-power batteries. Therefore, a key distribution scheme for WSNs must be an energy and memory efficient. In this poster, we proposed an energy-efficient hybrid key distribution scheme that is designed to suit the resource-constrained devices such as WSNs. We utilized Arduino UNO microcontroller and OPNET Modeler to investigate our proposed scheme in comparison to key distribution schemes in the literature. The findings show that our scheme achieves security and consumes less energy compared to other schemes in the literature. less energy.

Introduction

Recent advances in technology allow wireless sensor nodes to be cost-effective and small in size. Thus, they have become rapidly involved in a variety of applications such as in the military, health, agriculture, environment, home and commercial automation, and transportation. Key distribution plays a crucial role in the security of these applications. However, designing a key distribution scheme for WSNs is challenging because wireless sensor nodes are powered by limited-power batteries. In this poster, we present an energy-efficient hybrid key distribution scheme that is designed to suit resource-constrained devices such as WSNs.

Proposed Protocol

Pre–Deployment Phase:

$\{K_P, K_R\} \longleftarrow RSA_{Gen}$.

$K_P \overset{\text{def}}{=} AK_{sink}$ and $K_R \overset{\text{def}}{=} AK_{nodes}$.

$Sink\ node \coloneqq AK_{sink}$ and $Sensor\ node \coloneqq AK_{nodes}$.

Key Distribution Phase:

Sink node:

$K_{\text{session}} \overset{R}{\leftarrow} \{0,1\}^{128}$ and timestamp T.

$C \longleftarrow E_{AK_{sink}}(K_{\text{session}} \parallel T)$.

$\overset{send}{\Longrightarrow}[C \longleftarrow E_{AK_{sink}}(K_{\text{session}} \parallel T)]$.

Sensor nodes:

$\overset{rece}{\Longleftarrow}[C \longleftarrow E_{AK_{sink}}(K_{\text{session}} \parallel T)]$.

$P \longleftarrow D_{AK_{nodes}}\left(C \longleftarrow E_{AK_{sink}}(K_{\text{session}} \parallel T)\right)$.

$f_{\text{verif}}(T) = \begin{cases} \text{accept}, & T \le time\ threhsold \\ \text{reject}, & T > time\ threhsold \end{cases}$.

Post-Key distribution phase:

Sensor nodes:

$DataD$, and timestamp T.

$C \longleftarrow E^{\perp}_{K_{\text{session}}}(DataD \parallel T)$

$\overset{send}{\Longrightarrow}[C \longleftarrow E^{\perp}_{K_{\text{session}}}(DataD \parallel T)]$.

Sink node:

$\overset{rece}{\Longleftarrow}[C \longleftarrow E^{\perp}_{K_{\text{session}}}(DataD \parallel T)]$.

$P \longleftarrow D^{\perp}_{K_{\text{session}}}\left(C \longleftarrow E^{\perp}_{K_{\text{session}}}(DataD \parallel T)\right)$.

$f_{\text{verif}}(T) = \begin{cases} \text{accept}, & T \le time\ threhsold \\ \text{reject}, & T > time\ threhsold \end{cases}$.

Key Refreshment Phase:

Sink node:

$K_{New_{\text{session}}} \overset{R}{\leftarrow} \{0,1\}^{128}$ and timestamp T.

$C \longleftarrow E_{AK_{sink}}(K_{New_{\text{session}}} \parallel T)$.

$\overset{send}{\Longrightarrow}[C \longleftarrow E_{AK_{sink}}(K_{New_{\text{session}}} \parallel T)]$.

Sensor nodes:

$\overset{rece}{\Longleftarrow}[C \longleftarrow E_{AK_{sink}}(K_{New_{\text{session}}} \parallel T)]$.

$P \longleftarrow D_{AK_{nodes}}\left(C \longleftarrow E_{AK_{sink}}(K_{New_{\text{session}}} \parallel T)\right)$.

$f_{\text{verif}}(T) = \begin{cases} \text{accept}, & T \le time\ threhsold \\ \text{reject}, & T > time\ threhsold \end{cases}$

Methodology/implementation

We used real hardware implementation and simulation.

I. Hardware implementation.

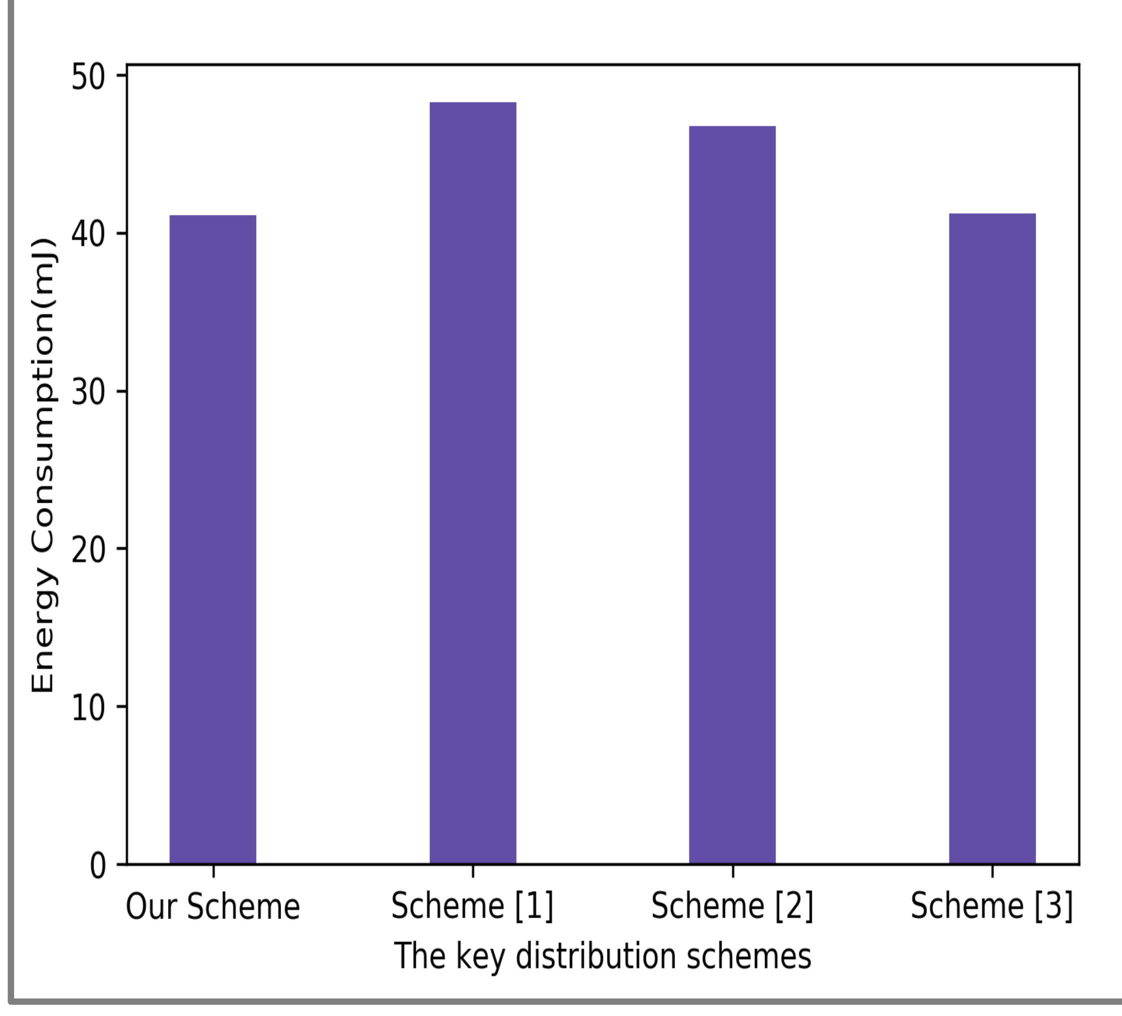
We used Arduino UNO microcontroller [1] to implement key distribution algorithms of the investigated schemes, measuring the time these schemes take to perform key distribution process and then calculating the actual energy consumption.

II. Simulation

We utilized OPNET Modeler to design and to create a model for a wireless sensor node. The model can calculate the energy consumption of a wireless sensor node as well as the energy consumption that is caused by wireless effects. The power parameters of our model is based on XBee transceiver S1 [2].

Efficiency Analysis

we examine the energy efficiency of our proposed scheme compared to the following key distribution schemes[3], [4], and [5]



Security Analysis

We utilized ProVerif, the automatic cryptographic protocol verifier to automatically analyze the security of our proposed scheme and verify it in a formal model. The Adversary model is based on Dolev-Yao model [6].

I. Reachability and Secrecy

Investigating the reachability of sensor *dataD* to an adversary.

```
Process:
{1}new senkey: prkey;
{2}let sinkey: pukey = pk(senkey) in
(
    {3}!
    {4}new sk_114: sessionKey;
    {5}new tsin: time;
    {6}let c: bitstring = encrypt((sk_114,tsin),sinkey) in
    {7}out(ch, c);
    {8}in(ch, y_115: bitstring);
    {9}let (dataDX: bitstring,tz: time) = sencrypt(y_115,sk_114) in
    {10}event acceptsSink(dataDX);
    {11}new tsin2: time;
    {12}if (tz = tsin2) then
    {13}event termSink(tz)
) | (
    {14}!
    {15}in(ch, x_116: bitstring);
    {16}let (sesk: sessionKey,tx: time) = decrypt(x_116,senkey) in
    {17}new tsen: time;
    {18}if (tx = tsen) then
    {19}event acceptsSensor(sesk);
    {20}let c': bitstring = sencrypt((dataD,tsen),sesk) in
    {21}out(ch, c');
    {22}event termSensor(sesk)
)

-- Query not attacker(dataD[])
Completing...
Starting query not attacker(dataD[])
RESULT not attacker(dataD[]) is true.
```

II. Correspondence Assertions

Sequence of events to model authentication.

```
Process:
{1}new senkey: prkey;
{2}let sinkey: pukey = pk(senkey) in
(
    {3}!
    {4}new sk_114: sessionKey;
    {5}new tsin: time;
    {6}let c: bitstring = encrypt((sk_114,tsin),sinkey) in
    {7}out(ch, c);
    {8}in(ch, y_115: bitstring);
    {9}let (dataDX: bitstring,tz: time) = sencrypt(y_115,sk_114) in
    {10}event acceptsSink(dataDX);
    {11}new tsin2: time;
    {12}if (tz = tsin2) then
    {13}event termSink(tz)
) | (
    {14}!
    {15}in(ch, x_116: bitstring);
    {16}let (sesk: sessionKey,tx: time) = decrypt(x_116,senkey) in
    {17}new tsen: time;
    {18}if (tx = tsen) then
    {19}event acceptsSensor(sesk);
    {20}let c': bitstring = sencrypt((dataD,tsen),sesk) in
    {21}out(ch, c');
    {22}event termSensor(sesk)
)

-- Query inj-event(termSink(x_117)) ==> inj-event(acceptsSink(y_118))
Completing...
Starting query inj-event(termSink(x_117)) ==> inj-event(acceptsSink(y_118))
RESULT inj-event(termSink(x_117)) ==> inj-event(acceptsSink(y_118)) is true.
-- Query inj-event(termSensor(x_326)) ==> inj-event(acceptsSensor(x_326))
Completing...
Starting query inj-event(termSensor(x_326)) ==> inj-event(acceptsSensor(x_326))
RESULT inj-event(termSensor(x_326)) ==> inj-event(acceptsSensor(x_326)) is true.
```

III. Observational Equivalence

The adversary could not acknowledge when the data$D$ get changed.

```
Process:
{1}new senkey: prkey;
{2}let sinkey: pukey = pk(senkey) in
(
    {3}!
    {4}new sk_114: sessionKey;
    {5}new tsin: time;
    {6}let c: bitstring = encrypt((sk_114,tsin),sinkey) in
    {7}out(ch, c);
    {8}in(ch, y_115: bitstring);
    {9}let (dataDX: bitstring,tz: time) = sencrypt(y_115,sk_114) in
    {10}event acceptsSink(dataDX);
    {11}new tsin2: time;
    {12}if (tz = tsin2) then
    {13}event termSink(tz)
) | (
    {14}!
    {15}in(ch, x_116: bitstring);
    {16}let (sesk: sessionKey,tx: time) = decrypt(x_116,senkey) in
    {17}new tsen: time;
    {18}if (tx = tsen) then
    {19}event acceptsSensor(sesk);
    {20}let c': bitstring = sencrypt((dataD,tsen),sesk) in
    {21}out(ch, c');
    {22}event termSensor(sesk)
) | (
    {23}phase 1;
    {24}new k: sessionKey;
    {25}new random: sessionKey;
    {26}out(ch, choice[random,k])
)

-- Observational equivalence
Termination warning: v_1455 <> v_1456 && attacker2_p1(v_1454,v_1455) && attacker2_p1(v_1454,v_1456) -> bad
Selecting 0
Termination warning: v_1460 <> v_1462 && attacker2_p1(v_1460,v_1461) && attacker2_p1(v_1462,v_1461) -> bad
Selecting 0
Completing...
Termination warning: v_1455 <> v_1456 && attacker2_p1(v_1454,v_1455) && attacker2_p1(v_1454,v_1456) -> bad
Selecting 0
Termination warning: v_1460 <> v_1462 && attacker2_p1(v_1460,v_1461) && attacker2_p1(v_1462,v_1461) -> bad
Selecting 0
Termination warning: v_2045 <> v_2046 && attacker(v_2045) && attacker2_p1(v_2045,v_2046) -> bad
Selecting 1
Termination warning: v_2074 <> v_2075 && attacker(v_2074) && attacker2_p1(v_2075,v_2074) -> bad
Selecting 1
RESULT Observational equivalence is true (bad not derivable).
```

Conclusion

Each time the distance double between two wireless sensor nodes, four times the amount of power are required. Therefore, proposing a key distribution scheme without considering the number of frames that are involved in key distribution process is not practical for such resource-constrained devices. Also, ignoring the wireless channel effects is not realistic because every wireless channel has effects that contribute to energy consumption. Therefore, when we designed our scheme, we consider all that issues.

References

[1] A. Coorporation, "Atmel ATmega328P Datasheet," ed, 2011.
[2] X. P. D. Sheet, "nd< www. sparkfun. com/ datasheets/ Wireless/Zigbee," *XBee-Datasheet. pdf*.
[3] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, 2005, pp. 524-535.
[4] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in Proceedings of the 9th ACM conference on Computer and communications security, 2002, pp. 41-47.
[5] Y. Zhang, "The scheme of public key infrastructure for improving wireless sensor networks security," in Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on, 2012, pp. 527-530.
[6] D. Dolev and A. Yao, "On the security of public key protocols," IEEE Transactions on information theory, vol. 29, pp. 198-208, 1983.