QR Code Based Mutual Authentication Protocol for Internet of Things

Tobias Marktscheffel
Institute of IT-Security
and Security Law (ISL)
University of Passau, Germany
tm@sec.uni-passau.de

Wolfram Gottschlich
Institute of IT-Security
and Security Law (ISL)
University of Passau, Germany
wg@sec.uni-passau.de

Wolfgang Popp
Institute of IT-Security
and Security Law (ISL)
University of Passau, Germany
poppwolf@fim.uni-passau.de

Philemon Werli
Institute of IT-Security
and Security Law (ISL)
University of Passau, Germany
werli@fim.uni-passau.de

Simon Dominik Fink
Institute of IT-Security
and Security Law (ISL)
University of Passau, Germany
finksim@fim.uni-passau.de

Arne Bilzhause
Institute of IT-Security
and Security Law (ISL)
University of Passau, Germany
ab@sec.uni-passau.de

Hermann de Meer
Department of Computer Networks
and Computer Communications
University of Passau, Germany
demeer@uni-passau.de

***Abstract*—In the Internet of Things (IoT), security is important and challenging; however, it is often neglected. This paper presents a smart home scenario, together with its requirements for a secure and user friendly mutual authentication protocol. Protocols developed for the internet are often not applicable to the Internet of Things due to hardware limitations and physical inaccessibility of devices. To tackle the challenge of a usable and secure device authentication in the area of the IoT, a QR code based mutual authentication protocol is proposed. The protocol supports two operation modes to handle different hardware configurations with respect to cameras and displays. Both operation modes are secure against attacks within the proposed attacker model. The protocol can also be used to exchange the public keys between two parties, in order to establish a secure channel without a trusted third party.**

***Index Terms*—Authentication, QR code, Internet of Things, Key Exchange, Usable Security**

I. Introduction

The Internet of Things (IoT) already affects our daily life and its impact will even increase in the future [1]. The range of application scenarios is huge and goes from smart homes, over smart cars and smart cities to smart spaces. Within these scenarios sensors generate private data, which is collected and processed further. In addition, actuators interact with the environment. This raises high and challenging requirements on the security and safety of such systems [2]. However, security and safety requirements are often only hardly considered or even not addressed at all.

A recent case, where security is not addressed at all, is the Nissan LEAF. There everyone can read data from a registered car via the internet, e.g. the length of the trips taken. In addition, an attacker can control the air condition. All this is possible mainly due to a lack of authentication. An attacker only needs the Vehicle Identification Number (VIN), which consists mainly of publicly available data like the brand and model of the car, besides this the VIN is displayed in the front shield of a car [3].

978-1-4799-8461-9/15/$31.00 ©2016 European Union

This points out the need to use secure authentication protocols. However, it is not trivial to find or develop suitable authentication protocols. A particular challenge is that many of the well known authentication and key exchange protocols, developed for the internet, are not applicable to the Internet of Things due to the fact that most services run on devices with limited hardware resources. The most relevant limitation factor with respect to the later described use case is the unavailability of hardware for interaction e.g. no keyboard, no mouse, if any only small screens. In addition, devices are often only physically accessible during the set-up phase or in case of maintenance work. This also embraces the usability aspect of security protocols for the IoT. Therefore, it is necessary to consider during design and implementation of protocols that they are suitable for different scenarios with specific hardware.

In this paper a QR code based mutual authentication protocol is described which was designed for a smart home scenario, but can also be used in various other scenarios. One such scenario is the authentication problem in the case of the above mentioned Nissan LEAF. In case of the later described smart home use case two different versions of the protocol are required, active and passive authentication. The necessity for this resulted in the fact that not all used devices support cameras and screens to read or display QR codes. This use case utilizes the protocol and shows the usability of it in order to secure a wireless communication.

The remainder of this paper is structured as follows. Section II describes the smart home use case. In Section III the considered attacker model is described. Then follows Section IV where relevant background information and the related work is provided, Section V describes the actual protocol. The paper concludes in Section VI with a discussion.

II. Use Case

The protocol was initially designed and implemented in a smart home system. In this smart home, three different types of devices exist, the base station (master), the sensor controlling devices (slaves) and user devices (smartphones). Slaves receive commands from their master and execute them, e.g. switching on a light. The smart home is based on a centralized architecture, with both smartphones and slaves communicating only with the master device. Slaves and sensors communicate directly with each other through a wired connection and thus this communication is assumed to be secure. Smartphones and slaves on the other hand are connected to the master via a wireless network or the internet, both are not seen as secure. In order to still be able to easily establish a secure connection over the wireless network, we have developed this protocol.

Before we explain the protocol, it is important to describe the hardware setup in further detail. The master and the slaves have a relatively small screen connected to themselves and every smartphone is equipped with a screen and a camera. With this hardware configuration we have the ability to display and scan QR codes and thus we can transfer small amounts of data between devices with a display and devices with a camera. This method of data transmission is assumed to be secure, because the scanning device must have physical access to the display to scan the QR code, which ensures that no malicious third party can intercept the QR code. Note that in our hardware setup only smartphones are equipped with a camera and thus can scan QR codes. To establish a connection with the master, slaves and smartphones need certain connection information, like the IP address. Instead of letting the user type in this information by hand, we encoded all this data in QR codes, which increases usability and avoids incorrect user input. To connect a new smartphone to the system, the user has to scan a QR code from the screen of the master, which contains all needed information to establish a connection. The structure of the QR codes will be explained further in Section V-A.

Slaves cannot establish a connection to the master the same way, because they cannot scan QR codes themselves. But by making use of the screen of the slave and an already secured connection between a smartphone and the master, it is still possible to connect the slave to the master. This is done by letting the master or a device, that has a camera attached and is connected to the master, scan the QR code that is displayed on the screen of the slave.

This leads us to two different versions of the protocol, called active and passive authentication. For the active authentication the new device scans a QR code. For the passive authentication the device that wants to be authenticated (new device) displays a QR code, which is scanned by a device that is communicating securely with the master. In the remainder of this paper, this device is often referred to as the ‘registered device’.

A graphical overview of both versions is given in Figures 1 and 2 and will be explained in more detail in Section V.

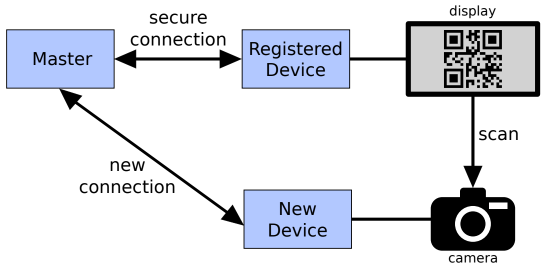


Fig. 1. Active scenario

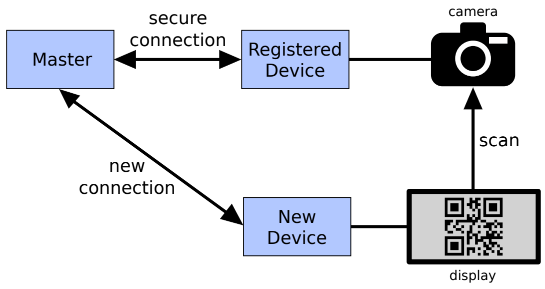


Fig. 2. Passive scenario

III. Attacker Model

In this paper we mainly consider a network-based attacker. Thus, we assume that the attacker has full access to messages exchanged via the network between the involved parties, i.e. the attacker can modify, delay, drop, and insert communication. In case the attacker drops messages, we do not guarantee the successful completion of our protocol any more. However, the attacker will still not be able to maliciously authenticate to the master. We also assume that the attacker cannot break the cryptography in any way, e.g. fake certificates.

Our approach relies on the QR code only being read by the camera of a legitimate device, and not by the attacker. Similarly, secret keys must be securely stored and not be accessed by an attacker. This implies that the attacker must not be able to access the memory of other processes which usually requires root access; an attacker only able to execute other apps on the systems (e.g. smartphones) would not break the security of our approach.

There have been many examples of privilege escalation attacks [4] on various platforms, including Android, on which we implemented our prototype. However, we consider the design of secure platforms for the Internet of Things to be another line of research, and therefore out of scope of this paper. Nevertheless, both secure platforms and secure communication between platforms are important for security in the Internet of Things. In this paper, we focus on secure communications.

IV. Background and Related Work

A. QR codes

The QR code is a two-dimensional barcode, which has become quite popular in recent years. QR codes are available in different sizes and error correction levels, which determine how much data can be stored in a QR code. This is standardized in ISO/IEC 18004 [5]. The size of a QR code is referred to as its version, which ranges from one to forty. QR codes are available in four different error correction levels to recover from loss of 7%, 15%, 25% or 30% of the symbol codewords. Since QR codes can only store textual data, a scheme for translating binary data to a string representation like *Base64* is required to store binary data.

While the structure of QR codes is too complex for humans, machines can read and decode a QR code easily. This makes QR codes very suitable for constrained devices, which are often used in IoT. Since many modern smartphones are equipped with camera and scanning software, the usage of QR codes will increase in the near future [6].

B. Related work

QR codes are being used for security-related purposes in various ways. Though they were not using QR codes specifically, the idea of using images and cameras for authentication was first discussed by Clarke et al. in 2002 [7]. Liao and Lee designed an authentication protocol using one-time passwords based on QR codes [8]; however, later Lee [9] showed that this protocol is vulnerable to a user impersonation attack. Similarly, Borisov [10] developed an authentication approach to login computer users into an industrial control system which relies on the user scanning a QR code with a smartphone on the respective component and laptop. All these approaches have in common that they aim toward a more secure login process. Unlike them, our approach uses the QR code to permanently register a new device to a server, i.e. improve security and usability during registration.

There are also some security issues associated with QR codes. As they are not human readable, the user is in general not able to discern if a QR code is malicious or not. This enables several types of attacks as described by Krombholz et al. [11], e.g. an attacker replacing the QR code being shown to a user with malicious content. Bani-Hani et al. [12] also discuss this attack vector and propose signing the data in QR codes and verifying the signatures with trusted third parties as a possible countermeasure. However, our approach is not vulnerable to these attacks for following reasons:

- Active Authentication: The QR code is displayed by a registered device. Since in our attacker model we assume all (registered) systems to be trustworthy, the QR code cannot be malicious.
- Passive Authentication: The QR code contains only token, address, and fingerprint. Assuming that our implementation is robust, the only error a malicious QR code can lead to is an authentication request being sent to a wrong address. However, the authentication will fail on every device except the one which generated the QR code.

Also in a more general Internet of Things context, without QR codes specifically, many researchers have recognized the challenges of authentication and identity management, leading to various related work: Different protocols using elliptic curve cryptography have been suggested [13], [14]. Mahalle et al. [15] combine this with capability-based access control. There is also substantial work on formal proofs for authentication protocols, e.g. Benin et al. for an authentication protocol using oblivious comparison [16]. On a higher level, Sarma et al. discuss concepts for identity management in the Internet of Things [17].

V. Protocol

In the following we give an insight on which information is encoded in the QR code and explain both the active and the passive versions of our protocol.

A. Structure of the QR code

The following information is encoded in the QR code:

- **Token**, an array of random bytes that is generated with a cryptographically secure pseudo-random number generator. The token is used to authenticate the device that scans the QR code
- **Address** of the device that generates the QR code
- **Fingerprint**, a cryptographic hash of the public key of the device that generates the QR code

The fingerprint is used to decrease the size of the QR code as the hash value of the public key is much smaller than the public key itself. Therefore, the size of the QR code is not affected by the length of the public key, but the length of the fingerprint. This makes the protocol independent from the used asymmetric cipher, like elliptic curve or RSA.

Now we give an example that explains, based on our smart home implementation, what size fingerprint, token and the resulting QR code have. The size of the fingerprint is 32 bytes, when using SHA256 to create the hash of the public key. As 128 bits of security are currently deemed secure enough against brute-force guessing attempts [18], the token should have a length of at least 16 bytes. The smart home is based on a IPv4 network, hence the IP-address has a fixed length of 4 bytes and the port number has a fixed length of 2 bytes. The fingerprint, token, IP-address and the port number sum up to 54 bytes, which can be encoded in a version 4 QR code with an error correction rate of 7%. When using *Base64* to convert bytes to a string representation, a version 4 QR code encodes up to 57 bytes[1], which leaves 3 bytes unused. These 3 bytes are used to increase the token length to 19 bytes, granting an even higher security level while optimally filling the available space in the version 4 QR code.

The sizes mentioned in the example above can be chosen differently. But note that the token has to be big enough and that you choose a cryptographic hash algorithm, that provides second pre-image resistance, so that the protocol, which is explained in the next two sections, stays secure.

[1]The size of a version 4 QR code is 33×33 pixels. The exact number of bytes that can be encoded depends on the *Base64* parameters.

B. Active Authentication

The description of the active Authentication in this section refers to Figure 3. Both Figure 3 and Figure 4 use the following notation:

$enc(k, msg)$:	encryption of message msg with the key k
$sig(k, msg)$:	signing of message msg with the key k
PKm:	the public key of the master device
SKm:	the private key of the master device
PKn:	the public key of the new device
SKn:	the private key of the new device
chl:	the challenge

The terms *master*, *slave* and *registered device* are explained in Section II together with a general overview over the authentication process. The separation of master and registered device is optional, the steps within the box (1 to 6) can also be performed on the master itself. If these devices are separated, a previously secured connection is required between the master and the registered device.

For active authentication, an already registered device first requests the QR code from the master (step 1). To answer this request, the master has to generate a new random token (step 2), which is saved for later validation (step 3). The master then generates the QR code as described in Section V-A (step 4). Finally, the master answers the initial request by sending the QR code through a previously secured connection (step 5). After receiving the QR code, the already registered device displays the QR code on its screen (step 6) waiting for the new device to scan the code.

After the new device has scanned the QR code (step 7), it extracts the connection information from the code (step 8). With this information the new device connects to the master and, while sending its own certificate, requests the certificate of the master (step 9). The master stores the received, currently still untrusted certificate (step 10) and responds by sending its own certificate to the new device (step 11), which validates it with the fingerprint that was part of the QR code (step 12). These certificates are used as container for the public keys for convenient storage and transportation. The master and the new device now verify that each other party holds the corresponding private key of the public keys they just exchanged. This is done by a mutual challenge-response authentication.

The mutual authentication sequence is initiated by the new device who sends a challenge to the master. The challenge is a randomly generated byte sequence used as nonce. This message is encrypted with the public key of the master (step 13). When the master receives the challenge, it decrypts the message with its own private key. The master creates the response of the challenge by signing the plain text challenge with its private key. The master now responds with a message that contains the response to the received challenge. After encrypting the whole message with the public key of the new device, the message is sent (step 14). When receiving the message, the new device first decrypts the message with its private key and then proceeds to verify the signature with the public key of the master. If the validation was successful, the new device compares the received response with the challenge that was sent to the master. If the comparison was successful, the new device can be sure that it is indeed communicating with the master that displayed the QR code (step 15). The new device can now authenticate itself to the master by sending the token that was extracted from the QR code. It first signs the token with its private key to verify its identity and then encrypts the signed token with the public key of the master. The message is sent to the master (step 16). After receiving the message, the master decrypts the message with its private key. The master then validates the signature of the sent token with the public key of the new device. When the validation is successful, the master compares the token with the saved token from step 3. If both are equal, the master can be sure that the new device is indeed the device that scanned the QR code (step 17). To prevent further reuse the master deletes the token (step 18).

At this point the connection between the master and the new device is established, the public keys are exchanged and the devices are mutually authenticated.

The master can now send feedback to both the new device and the registered device (steps 19 & 20).

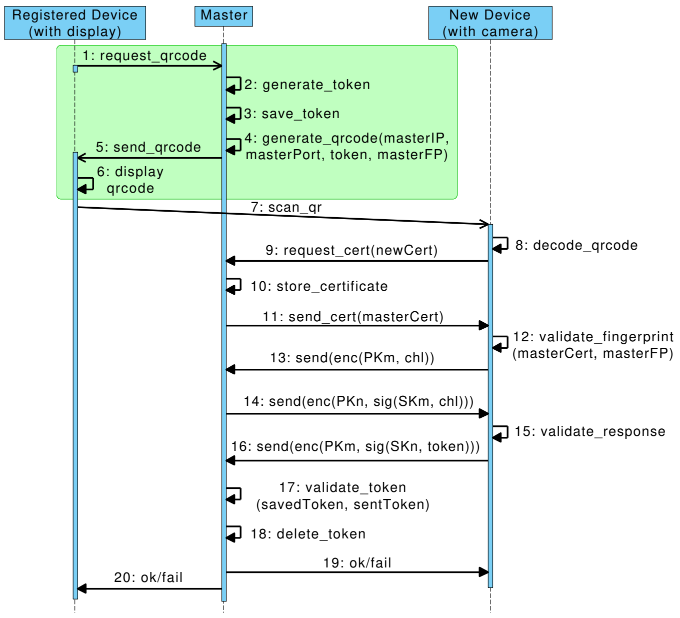

Registered Device (with display)
Master
New Device (with camera)
1: request_qrcode
2: generate_token
3: save_token
4: generate_qrcode(masterIP, masterPort, token, masterFP)
5: send_qrcode
6: display qrcode
7: scan_qr
8: decode_qrcode
9: request_cert(newCert)
10: store_certificate
11: send_cert(masterCert)
12: validate_fingerprint (masterCert, masterFP)
13: send(enc(PKm, chl))
14: send(enc(PKn, sig(SKm, chl)))
15: validate_response
16: send(enc(PKm, sig(SKn, token)))
17: validate_token (savedToken, sentToken)
18: delete_token
19: ok/fail
20: ok/fail

Fig. 3. Active Authentication

C. Passive Authentication

The following description refers to Figure 4.

For passive authentication the new device generates and saves an authentication token (steps 1 & 2). After generating the token, the new device generates the QR code as described in Section V-A (step 3). The QR code is now displayed on the screen directly connected to the new device (step 4).

After an already registered device is trigged by a user, it scans the QR code with its camera (step 5) and extracts the token and information on how to connect to the new device (step 6). The extracted information is then forwarded through a previously secured communication channel to the master (step 7). If the master has a camera connected to it, steps 5 to 7 can also be performed on the master itself, indicated by the highlighted rectangle in Figure 4.

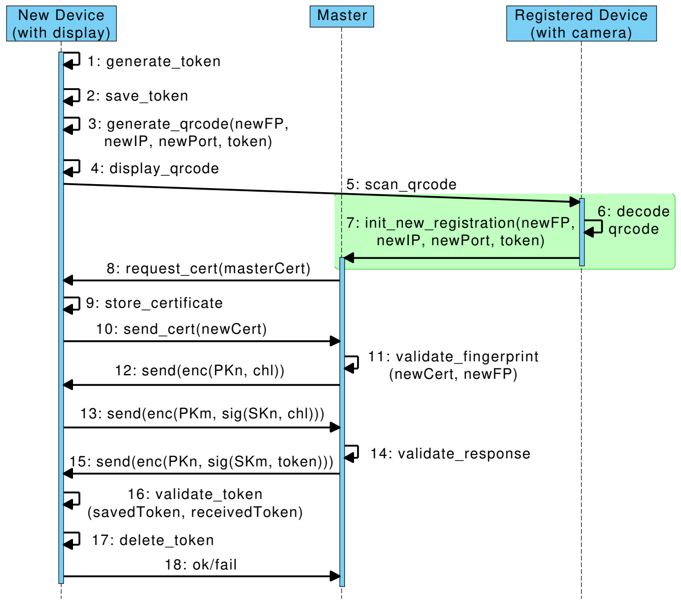
With this information the master connects to the new device and, while sending its own certificate, requests the certificate of the new device (step 8). The new device stores the received, currently still untrusted certificate (step 9) and responds by sending its own certificate to the master (step 10), who validates it with the fingerprint that was part of the QR code (step 11). The master and the new device now verify that each other party holds the corresponding private key of the public keys they just exchanged. This is done by a mutual challenge-response authentication.

The mutual authentication sequence is initiated by the master who sends a challenge to the new device. The challenge is a randomly generated byte sequence used as nonce. This message is encrypted with the public key of the new device (step 12). When the new device receives the challenge, it decrypts the message with its own private key. The new device creates the response of the challenge by signing the plain text challenge with its private key. The new device proceeds to encrypt the signed response with the public key of the master and sends the message to it (step 13). After receiving the message, the master first decrypts the message with its private key and then proceeds to verify the signature with the public key of the new device. If the validation was successful, the master compares the received response with the challenge that was sent to the new device. If the comparison was successful, the master can be sure that it is indeed communicating with the device that displayed the QR code (step 14). The master now authenticates itself to the new device by sending the token that was extracted from the QR code. It first signs the token with its private key to verify its identity and then encrypts the signed token with the public key of the new device. The message is sent to the new device (step 15). After receiving the message, the new device decrypts it with its private key. The new device then validates the signature of the sent token with the public key of the master. When the validation is successful, the new device compares the token with the saved token from step 2. If both are equal, the new device can be sure that the master is indeed the master of the registered device that scanned the QR code (step 16). To prevent further reuse the new device deletes the token (step 17).

At this point the connection between the master and the new device is established, the public keys are exchanged and the devices are mutually authenticated. The new device can now send feedback to the master (step 18).

VI. Discussion

We will now evaluate the security of our active and passive authentication protocol, followed by the conclusion.



New Device (with display)
Master
Registered Device (with camera)
1: generate_token
2: save_token
3: generate_qrcode(newFP, newIP, newPort, token)
4: display_qrcode
5: scan_qrcode
6: decode qrcode
7: init_new_registration(newFP, newIP, newPort, token)
8: request_cert(masterCert)
9: store_certificate
10: send_cert(newCert)
11: validate_fingerprint (newCert, newFP)
12: send(enc(PKn, chl))
13: send(enc(PKm, sig(SKn, chl)))
14: validate_response
15: send(enc(PKn, sig(SKm, token)))
16: validate_token (savedToken, receivedToken)
17: delete_token
18: ok/fail

Fig. 4. Passive Authentication

A. Security Evaluation

An attacker with the capabilities described in the attacker model would be able to perform *man in the middle attacks* (MITM). However, such an attack is not successful in the proposed protocol. An attacker cannot pretend towards the scanning device to be the device that showed the QR code since he verifies its identity (public key) with the help of the fingerprint. An attacker also cannot pretend towards the device that showed the QR code to be the scanning device because he verifies the token which is not known to an attacker. This holds since the token is only transmitted via the QR code and encrypted with the verified public key of the device that showed the QR code and signed by the device that scanned the QR code. Therefore, a MITM-attack cannot be successfully realized for the active and passive version of the protocol.

Our countermeasures against *replay attacks* are based on randomly generated tokens and challenge-response as well as the QR codes. In passive authentication, an attacker can replay steps 1–4; he can also possibly generate a token which was previously used. However, in step 5, the attacker would need to convince a registered user to scan another QR code, i.e. for the user to take a registered device with camera and scan a QR code on a device controlled by the attacker. This would raise suspicions on an already registered device.

In active authentication, a replay attacker can reproduce the protocol starting in step 9. He can replay the execution of the protocol until step 16. In step 16, he has to know the token which was encoded in the QR code. Therefore, *reflection attacks* on the challenge are also not successful because our protocol additionally relies on the token contained in the QR code which cannot be reflected. Furthermore, replay attacks would only be successful, if the attacker were able to secretly scan the QR code[2]. As a countermeasure, the master deletes the token in step 18 and remembers the state of a protocol run, i.e. expects step 9 only once after it generated a QR code.

Hence, once active authentication is completed, an attacker cannot gain access by replaying (parts of) the protocol run, i.e. active authentication is not vulnerable to replay attacks.

Alternatively, an attacker could try to exploit a race condition: when active authentication is started, the attacker captures the QR code, and executes the active authentication protocol faster than the benign new device. The attacker can suppress the messages originating from the benign device that is to be authenticated to ensure that he is faster than the benign device. However, a reasonable user would notice that his new device was not able to complete authentication. Thus, the legitimate user will immediately remove the attacker and rerun authentication with his new device.

Similarly, during passive authentication, an attacker with the scanned QR code could exploit a race condition starting with step 8: The attacker could temporarily impersonate the master towards the new device, and complete registration. However, the user would again immediately notice that the new device was not successfully registered to the master, and discard the relationship established with the attacker.

The attacker can also attempt *denial-of-service* attacks on the communication between master and another device. However, this can only prevent the successful completion of the protocol, but does not help the attacker in any way to gain unauthorized access. In addition, only registered devices can make computationally expensive requests to the master, e.g. generating a QR code; thus, an attacker cannot easily bring the master out-of-service with malicious requests.

B. Conclusion

The mutual authentication protocol introduced in this paper is applicable to a wide range of use cases in the area of IoT, mainly because of its ease of use that is obtained through the machine-readable QR codes. These use cases include the described smart home system as well as the smart car example given in Section I. The protocol is also applicable to any type of network based setup that assumes the same attacker model as described in Section III and where devices are equipped with a screen or camera during deployment. As future work we want to adopt the implementation of the smart home scenario for other platforms to further utilise the protocol.

Additionally, the public keys that are exchanged via this protocol can be stored and used to encrypt future communication or to exchange symmetric keys. However, when saving the public keys, key management including revocation has to be addressed, which is not part of our protocol.

As shown in the security analysis, the protocol is not vulnerable to the described attacker without physical access. Hence, the described approach provides the possibility to improve security and to maintain a high usability in IoT environments at the same time.

[2]which the attacker cannot according to our attacker model, cf. Section III

ACKNOWLEDGEMENTS

The research leading to these results was supported by the Bavarian State Ministry of Education, Science and the Arts as part of the FORSEC research association. In addition, we would like to thank Andreas Bucher, Christoph Frädrich, and Leon Sell who helped to develop the original version of the prototype and Wolfgang Duschl and Martin Bochenek for their feedback during the development of the protocol.

REFERENCES

[1] A. Castillo and A. D. Thierer, "Projecting the growth and economic impact of the internet of things," *Available at SSRN 2618794*, 2015.

[2] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: the internet of things architecture, possible applications and key challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference on*, pp. 257–260, IEEE, 2012.

[3] "Controlling vehicle features of nissan leafs across the globe via vulnerable apis." http://www.troyhunt.com/2016/02/controlling-vehicle-features-of-nissan.html. Accessed: 2016-01-03.

[4] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A.-R. Sadeghi, and B. Shastry, "Towards taming privilege-escalation attacks on Android," in *19th Annual Network & Distributed System Security Symposium (NDSS'12)*, Feb 2012.

[5] I. O. for Standardization, "Information technology – automatic identification and data capture techniques – qr code bar code symbology specification." http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=62021. Accessed: 2016-07-03.

[6] "Gartner identifies china's top 10 strategic technology trends in 2014." http://www.gartner.com/newsroom/id/2764217. Accessed: 2016-09-03.

[7] D. E. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. L. Rivest, "The untrusted computer problem and camera-based authentication," in *Pervasive Computing, First International Conference, Pervasive 2002, Zürich, Switzerland, August 26-28, 2002, Proceedings*, pp. 114–124, 2002.

[8] K.-C. Liao and W.-H. Lee, "A novel user authentication scheme based on qr-code.," *JNW*, vol. 5, no. 8, pp. 937–941, 2010.

[9] Y. Lee, J. Kim, W. Jeon, and D. Won, *Information Technology Convergence, Secure and Trust Computing, and Data Management: ITCS 2012 & STA 2012*, ch. Design of a Simple User Authentication Scheme Using QR-Code for Mobile Device, pp. 241–247. Dordrecht: Springer Netherlands, 2012.

[10] A. Borisov, "A novel approach for user authentication to industrial components using qr codes," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 3, pp. 61–66, July 2015.

[11] K. Krombholz, P. Fruehwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl, *QR Code Security: A Survey of Attacks and Challenges for Usable Security*, pp. 79–90. Springer, 0 2014.

[12] R. M. Bani-Hani, Y. A. Wahsheh, and M. B. Al-Sarhan, "Secure qr code system," in *Innovations in Information Technology (INNOVATIONS), 2014 10th International Conference on*, pp. 1–6, Nov 2014.

[13] J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and access control in the internet of things.," in *ICDCS Workshops*, pp. 588–592, IEEE Computer Society, 2012.

[14] G. Zhao, X. Si, J. Wang, X. Long, and T. Hu, "A novel mutual authentication scheme for internet of things," in *Modelling, Identification and Control (ICMIC), Proceedings of 2011 International Conference on*, pp. 563–566, June 2011.

[15] P. N. Mahalle, B. Anggorojati, N. R. Prasad, and R. Prasad, "Identity establishment and capability based access control (iecac) scheme for internet of things," in *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, pp. 187–191, Sept 2012.

[16] A. Benin, S. Toledo, and E. Tromer, "Secure association for the internet of things.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 940, 2015.

[17] A. C. Sarma and J. Girão, "Identities in the future internet of things," *Wireless Personal Communications*, vol. 49, no. 3, pp. 353–363, 2009.

[18] E. Barker, "Recommendation for key management part 1: General, revision 4," Tech. Rep. 800-57 Part 1, National Institute of Standards and Technology, January 2016. Page 54-56.