Hardware Secured, Password-based Authentication for Smart Sensors for the Industrial Internet of Things

Thomas W. Pieber[1], Thomas Ulz[1], Christian Steger[1], and Rainer Matischek[2]

[1] Graz University of Technology - Institute for Technical Informatics, Graz, Austria
{thomas.pieber, thomas.ulz ,steger}@tugraz.at
[2] Infineon Technologies Austria AG, Graz, Austria
rainer.matischek@infineon.com

Abstract. Sensors are a vital component for the *Internet of Things*. These sensors gather information about their environment and pass this information to control algorithms and/or actuators. To operate as effective as possible the sensors need to be reconfigurable, which allows the operators to optimize the sensing activities. In this work we focus on the mechanisms of such reconfiguration possibilities. As the reconfiguration can also be used to manipulate the sensors (and their attached systems) in a subtle way, the security of the reconfiguration interface is of utmost importance. Within this work we test a lightweight authentication method for use on a smart sensor and describe a possible implementations of the authentication mechanism on a hardware security module.

1 Introduction

In the Internet of Things (IoT) sensors are key components. They create the bulk of the information needed to control their environment according to the wishes of the operator. They furthermore monitor the environment and need to make decisions if the monitored environment is changing in a way that needs the operators' attention. The sensors are equipped with some sort of microcontroller, software, energy source, communication mechanism, and most likely an interface for configuring the software and the decision making. This interface poses a threat to the integrity and trustworthiness of the sensor itself and, in the long run, to the whole system. In order to become a trustworthy system the configuration- and sensor data must be protected against all adversaries. To accomplish this, the sensors can be equipped with tamper resistant hardware security modules (hereafter HSM or security controller). This HSM on a smart sensor can perform critical operations during the configuration of the device and during communication of sensor data to the outside world. These critical operations include the encryption of sensor data, establishing a secured and authenticated channel to maintenance personnel, and the secured storage of configuration- and authentication data and cryptographic keys.

The authentication of users that can see and manipulate the confidential settings of the sensor is one of the core features that a secured system needs. This process not only blocks others from accessing the confidential information but also enforces that only trusted personnel can manipulate the settings. One of the most convenient and widely used methods of authentication is the use of passwords. These passwords can be remembered by the trusted operators and take the function of a shared secret. In a conventional system the user shows the knowledge of the shared secret by directly entering it on the used device. In the case of remote authentication, as it is the case with smart sensors, the password must be transmitted securely to the verifying party. In an unconstrained device this would be accomplished by securing the channel against eavesdroppers and then sending the password over the secured channel. For resource constrained devices this method is impractical. Therefore, methods that perform the authentication alongside the establishment of the secured channel have been developed. This is not only faster than performing these operations sequentially, but also more efficient in terms of energy usage, computation steps needed, and memory allocated on the constrained device. At this step an HSM with dedicated cryptographic hardware can also perform these steps faster, more efficient, and in a more secured fashion compared to a normal microcontroller. The HSM can additionally store the users' credentials in the tamper resistant memory, keeping information leakage as low as possible.

A HSM is typically very limited in terms of general computational power and available memory. This entails that the used protocols must be lightweight in those parameters. This is additionally challenged by the energy constraints on the smart sensor. As such sensors might be operated using battery power, the cryptographic challenging (and therefore energy hungry) functions must be reduced to a minimum to keep the sensor alive as long as possible.

In this work we examined the use of a lightweight authentication method for smart sensors. Therefore, we used simulation techniques to design and implement a prototype application and tested the results on an Infineon type security controller.

The remainder of this paper is structured as follows: In Section 2 the prerequisites for the implementation and related works are stated. The design questions and the approaches are elaborated in Section 3. Details on the implementation are given in Section 4. The 5^{th} section is dedicated to the evaluation of the authentication protocol and answers why the used protocol and hardware is suitable for smart sensors. Possible future work is stated in Section 6. The paper concludes with Section 7.

2 Related Work

To perform a secured authentication a key agreement protocol needs to be used. the most widely used protocol for this task is the Diffie-Hellman key exchange [1]. This protocol defines public and private keys. After the exchange of the public keys a shared secret can be calculated. The security of that scheme is based on the Computational Diffie-Hellman problem. However, this protocol alone is not able to authenticate the communicating parties.

The authentication of users is a crucial part of a secured communication as an adversary can impersonate the other communication partner and perform a man-in-the-middle attack. Typically, the authentication is done with a shared secret - a password. There are many algorithms that use passwords for authentication. One of them is proposed by M. Peyravian and N. Zunic [2]. This protocol is especially well-suited for use in microcontrollers as the only cryptographic function is a collision-resistant hash function H and therefore uses very little computational effort. In this protocol the user sends his username (un) and a nonce (ru) to the server which replies with another nonce (rs). The user, knowing the password (pw), calculates the function M = H(H(un, pw), ru, rs) and sends the resulting M to the server. The server then uses a lookup-table to get the H(un, pw) matching to the username and can verify M. This protocol can authenticate a user against the server but does not provide the needed security on its own. This has to be done beforehand with a key exchange. This, on the other hand, needs more computational time as the two protocols need to be executed. Another, newer protocol was proposed by I. Liao, C. Lee, and M. Hwang [3]. In this protocol a message pair is exchanged during the registration and further three messages need to be sent for the authentication. This protocol, when executed with ECC, requires four multiplications, two additions, five hash-operations and one random number. Additionally to those operations, the operations for securing the channel in the first place have to be added.

There are also protocols that perform a key agreement while authenticating the user to the server. One of the first protocols that perform an authenticated key exchange is the EKE protocol proposed by S. M. Bellowin and M. Merrit proposed [4]. This protocol is the predecessor to most modern protocols. It works by symmetrically encrypting a public key and session key with the password. This is also the weak part of this protocol as S. Halevi and H. Krawczyk [5]. They say that it is not wise to use a password (or any other low-entropy key) as a key to a cryptographic function. The term “low-entropy” means that even a random string of ASCII-characters uses only the letters from 20 to 126 (=106) of all 256 possible 8-bit characters. This means that more than half of the possibilities are not used and therefore such strings should not be used for encryption, only for authentication.

There are many variants of this protocol in the literature such as [6–10]. The Gennaro-Lindell PAKE protocol from [10] can be proven secure in the standard model of cryptography. A computationally less expensive protocol is the SPAKE-protocol from [6]. This protocol was proposed by M. Abdalla and D. Pointcheval and is proven to be secure under the random oracle model. It is also very efficient as it only needs two messages to be exchanged. The whole protocol, when implemented with ECC, uses only six multiplications, two additions, five hash-operations, and one random number. Abdalla states in [11] that:

> [. . .] the simple password-authenticated key exchange protocol [. . .] to which we refer as SPAKE [. . .] is among the most efficient PAKE schemes based on the EKE protocol.

3 Design

There are two important questions to be answered in order to design an authentication mechanism for smart sensors. (I) In which situation of the sensor's lifecycle is this authentication going to be used? (II) What hardware can be used in order to perform the critical steps - and what costs / benefits come with that hardware?

The answers to these questions are intertwined. The configuration of the sensor system should be able to be performed at any time during the sensor's lifecycle. That means that the configuration interface might not be connected to any power source or even the controller storing the configuration data is not connected at all (e.g. during the production of the system parts). This leads to the use of Near Field Communication (NFC) as a communication mechanism and energy source for the chip containing the (confidential) parameters. This entails that a security controller that is capable of using an NFC antenna is needed. One controller that is capable of such operation is used in [12]. The use of this controller furthermore comes with the benefit that the cryptographic functions, necessary during the communication, can be performed with the harvested energy and do not consume the limited energy source on board of the sensor.

Another requirement that comes from using the same authentication method throughout the entire lifecycle of the sensor, is the possibility of changing cryptographic keys, encryption parameters, usernames and passwords. This can be subsumed with the term *Bring Your Own Key / Encryption (BYOK / BYOE)* [13]. With the use of such methods it can be assured that the device cannot be read by anyone except the authorized persons. This rises the need for a secure user authentication scheme. The authentication of the sensor itself can be performed by only sharing the password with one only one sensor.

The SPAKE-protocol introduced in [6] is a suitable protocol for establishing an authenticated secure link between the trustworthy sensor and the user. This protocol performs a Diffie-Hellman key agreement to generate the session key for the encrypted messages. To authenticate the user a mask is calculated that is applied to the public keys of both partners. At the remote end the mask is removed and the key agreement finishes. If the calculated masks do not match the resulting key will be different and the communication cannot be performed.

Because of the reduced memory consumption on the smart sensor and the sufficient hardware acceleration of the secure element, elliptic curve cryptography (ECC) was chosen to perform the key agreement and authentication. In Table 1 the SPAKE2 algorithm using ECC is shown. There $K_A \overset{!}{=} K_B$ and therefore secret symmetric the key $sk_A = sk_B$.

Table 1. Design of the ECC implementation of the SPAKE2 algorithm [6]

public information: $G, H(\cdot), u_A, u_B$

private shared information: $password$

User A		User B
$x = rand()$		$y = rand()$
$X = xG$; $M = H(u_A)G$		$Y = yG$; $N = H(u_B)G$
$X^* = X + (password)M$		$Y^* = Y + (password)N$
	$X^* \rightarrow$	
	$\leftarrow Y^*$	
$N = H(u_B)G$		$M = H(u_A)G$
$K_A = x(Y^* + Inv((password)N))$		$K_B = y(X^* + Inv((password)M))$
$sk_A = H(H(u_A), H(u_B), X^*, Y^*, password, K_A)$		
		$sk_B = H(H(u_A), H(u_B), X^*, Y^*, password, K_B)$

The use of a special hardware for the authentication yields some constraints on what operations can be used to perform the authentication. Such constraints can come from be the available memory of the HSM, the computational speed, the energy intake during the computation, and the hardware support of cryptographic functionalities.

The selected HSM supports ECC operations up to 256 bits and SHA256. Therefore, the final implementation only supports curves with 256-bit parameters and uses the SHA256 algorithm to perform the final key generation and key expansion for the username and password. The constraints on the memory entail a maximum of different curves and users.

3.1 Evaluation Design

The evaluation of the implementation against the reference from Googles Weave project [14] cannot be performed, as currently fundamental differences between that implementation and the description of the protocol in [6] exist. Most notably Googles design uses fixed points for M and N and they do not use a Key-Derivation-Function to generate the session key - this is a mistake denoted in [6] that would render the authentication insecure. Furthermore, Google uses 224-bit parameters while this implementation of SPAKE2 uses 256-bit. Therefore, the evaluation will concentrate on the performance of our implementation of the SPAKE2-protocol on the HSM.

To evaluate the performance of the protocol the security controller communicates with an implementation on a laptop computer over an NFC interface. To get a complete picture of the protocol-performance the round-trip time for packets with different lengths is the baseline. For further testing a version of the SPAKE2 protocol was deployed on an NFC enabled Android device. There the whole process for authentication and writing sample configurations was monitored.

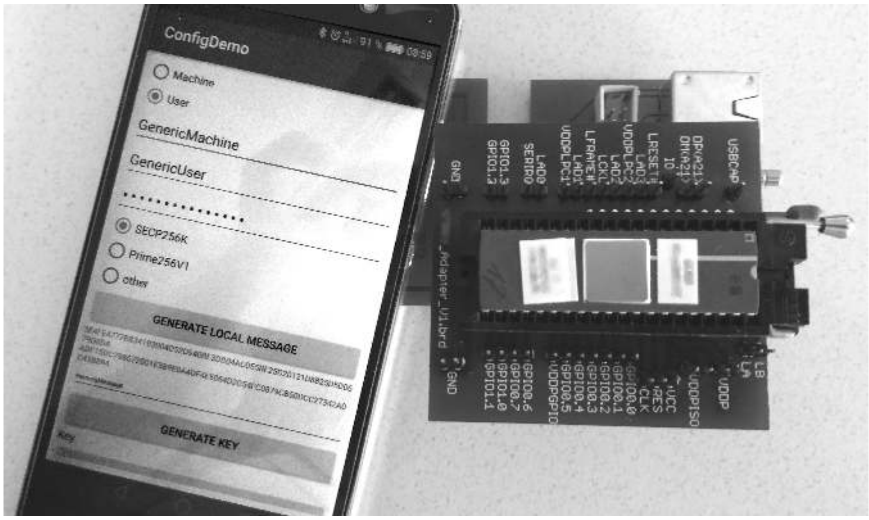

Fig. 1. Prototype-hardware used for evaluating the performance

For further comparison, two variants of the protocol are implemented on the security controller. One that is optimized for low memory usage and one that has low computing times at the cost of increased memory consumption. These two versions are then compared against each other to evaluate the performance change if the cryptographic operations for authentication are reduced to a minimum. To demonstrate the capabilities of using different curves and users two curves and users should be supported.

4 Implementation

To test the implementation on the HSM a command structure based on Application Protocol Data Units (APDUs) has been defined. To be able to test the performance of the single steps of the protocol a command for every operation was defined. The operators' device sends the necessary commands to the HSM, retrieves the answers from the HSM, takes timing measurements, and asks the operator for the desired operation and, if necessary for the authentication credentials.

Figure 1 shows the authentication interface on the NFC-enabled smartphone asking the operator for the credentials. This test version is configured to emulate a user and has the demo credentials entered. The implementations support the handling of arbitrary ECC curves; it therefore has a setting to insert the curve parameters manually. The buttons visible can generate the local public key and generate the final key after the communication is complete. After the authentication is finished successfully the operator can view the device's configuration, alter the received configurations, and deploy an altered one.

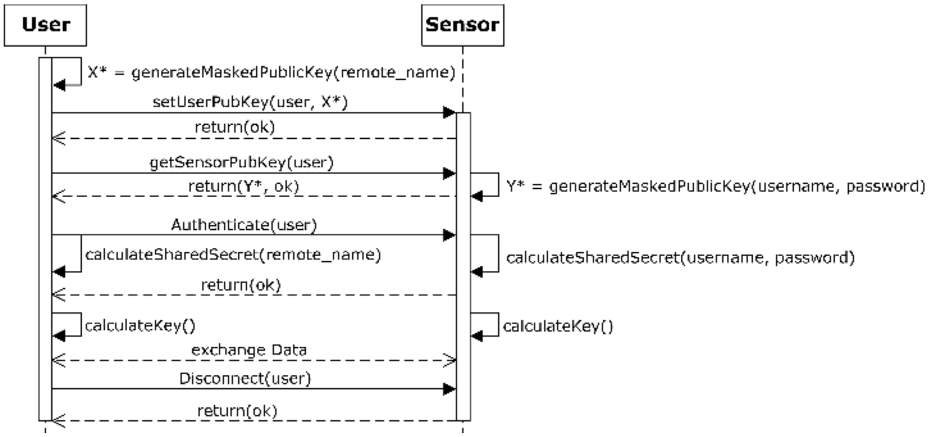

Fig. 2. Communication structure for fine granular evaluation

5 Performance Evaluation

The measurement of the bare communication with different sized payloads is performed with the echo command. If the user is authenticated this command returns the payload, otherwise an empty packet with an error number is returned. As expected, the time used for communication increases linearly with the payload length. Also the time for communication approximately doubles if the payload is also sent back. When the corresponding time is subtracted of the communication time of other commands, it can be evaluated how long the operations on the secure element take to perform.

As expected, the timing of the authenticated echo request is slightly more than double of the unauthenticated one. This is because the data has to be sent back again and some more internal computation has to be done.

In Figure 2 the communication structure for evaluation purposes is shown. It comprises of the calculations on the two sides of the communication and the communication itself.

The average timing of generating the public key on the HSM is about 218800 μs. Considering that the communication of one empty packet and one packet with 64 bytes payload takes on average 8000 μs, one can conclude that the necessary calculations can be performed in 210800 μs or about 210 ms. The key derivation (K_A and K_B) also takes about 200 ms. The calculation of the key from the users credentials and the shared secret uses approximately 37 ms. All those measurements were made 1000 times. The distribution of the values is gaussian. The gaussian parameters for the operations can be found in Table 2. With those numbers we can estimate that the whole authenticated key exchange can be performed in about 426 ms. The time to initialize the users is negligible as only the credentials need to be saved. Other methods like combining ECDH and the protocol proposed in [3] take approximately an equal amount of operations on every run but require additional messages to be sent and perform cryptographic

Table 2. Timings of the different operations

Operation (low-mem)	Mean [μs]	Sigma [μs]
unauthSend(64b)	8023	45
sendKey	8295	61
generatePubKey	218813	205
calculateSharedSecret	203548	142
calculateKey	42381	68
Operation (low-comp)	**Mean [μs]**	**Sigma [μs]**
generatePubKey	74077	67
calculateSharedSecret	67858	125
calculateKey	39554	49
initUser	91002	28
initMachine	12453 + 43542/User	18
changeMachine	273919	170

functions on initializations of new users. A comparison of the necessary operations is shown in Table 3.
If the operations are altered in a way such that more computations are done when initializing a new user (or altering the user-credentials) it would take more memory space but the computation time on every run can be cut down to about 145 ms on every run and 91 ms upon initializing the user. Changing the credentials of the security controller requires that some credentials of the users need to be recalculated. This requires about 110 ms per user. These numbers are shown combined in Table 2 in the *low-comp* section.

Table 3. Comparison between different authentication schemes.

	ECDH	[3]	SPAKE-low-mem	SPAKE-low-comp
Crypto-operations $(*; +; H(\ldots); rand())$	2; 0; 1; 1	3; 2; 6; 1	6; 2; 5; 1	2; 2; 1; 1
initialization crypto-operations	0; 0; 0; 0	3; 2; 0; 0	0; 0; 0; 0	4; 0; 2; 0
Time authentication [ms]	120	279 + ECDH	426	145
Time initialization [ms]	0	203	0	~100
Permanent Memory / User	0	credentials	credentials	2 Hashes + 2 Points
Permanent Memory SE	0	credentials	credentials	1 Hash

Table 3 compares two possible SPAKE implementations (one designed for low memory usage, one for low computation time) with operations needed when using ECDH and the authentication scheme proposed by [3]. It furthermore shows approximated timings for the operations when executed on the chosen HSM. The protocol from [3] uses a previously secured communication channel to transmit data. Therefore, an algorithm like ECDH needs to be executed before the authentication is performed. The table shows the required cryptographic operations for every authentication and the operations necessary when initializing a new user. Furthermore, the timings for initializing and authentication on the HSM, and the memory usage for the different algorithms per user and for the HSM are shown. The comparison indicates that the SPAKE protocol is at least as efficient as the protocol proposed by I. Liao, C. Lee and M. Hwang [3]. If the hardware allows for more memory usage, the shown SPAKE2 implementation performs the authentication on top of the ECDH with little overhead.

With these statistics the strength of the implemented SPAKE2 protocol gets visible. While other protocols initially perform a key exchange followed by the authentication, these two steps get done in a single operation. Not only the time used to communicate, but also the time needed to protect the authentication data during transport can be reduced. With this the transmission of data can stop earlier, resulting in decreased energy consumption. Additionally, the amount of required cryptographic operations is reduced. A protocol using an ECDH scheme and an authentication protocol afterwards uses more random numbers, more hash-operations, and more ECC-related functionalities. Furthermore, the memory usage per user can be just a few bytes (the credentials and additional information for authorization and cryptography details). The term *credentials* means that the password and username are stored in plain text inside the tamper resistant memory. In the low-comp section, the credentials are stored in their key-expanded form (*Hash*) as they will be used like this during the computation. Additionally the ECC points M and N are calculated beforehand and stored in the memory to reduce computation time.

6 Future Work

As previously described, the authorization and establishing of the secured channel can be performed with two messages. This can reduce the communication overhead, which is especially useful for low-powered sensors where the use of the communication devices consumes most of the available energy. This protocol can be changed to also enable authentication between machines. This is especially useful for an industrial setting where robots need to communicate with other machinery to fulfil their tasks. It can also be combined with other communication techniques to enhance current sensor configuration possibilities.

7 Conclusion

In this paper an implementation of the SPAKE2-algorithm [6] has been shown. The evaluation shows that the protocol can be implemented efficiently on an HSM. It also shows that the use of an authenticated key agreement protocol is advantageous compared to a standard solution where key agreement and authentication are performed separately. These features naturally reduce the communication overhead and can be implemented with little overhead in computation or memory size. Combined, this leads to the conclusion that an authenticated key agreement like SPAKE2 is useful for the use on a smart sensor.

Acknowledgment

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Germany, Netherlands, Spain, Austria, Belgium, Slovakia.

References

1. Diffie, W., Hellman, M.: New directions in cryptography. IEEE transactions on Information Theory **22**(6) (1976) 644–654
2. Peyravian, M., Zunic, N.: Methods for protecting password transmission. Computers & Security **19**(5) (2000) 466–469
3. Liao, I.E., Lee, C.C., Hwang, M.S.: A password authentication scheme over insecure networks. Journal of Computer and System Sciences **72**(4) (2006) 727–740
4. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on, IEEE (1992) 72–84
5. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Transactions on Information and System Security (TISSEC) **2**(3) (1999) 230–268
6. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Cryptographers Track at the RSA Conference, Springer (2005) 191–208
7. Bellare, M., Rogaway, P.: The autha protocol for password-based authenticated key exchange. Technical report, Technical report, IEEE (2000)
8. Kobara, K., Imai, H.: Pretty-simple password-authenticated key-exchange under standard assumptions. iacr eprint archieve, 2003
9. Krawczyk, H.: Sigma: The sign-and-macapproach to authenticated diffie-hellman and its use in the ike protocols. In: Annual International Cryptology Conference, Springer (2003) 400–425
10. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2003) 524–543
11. Abdalla, M.: Password-based authenticated key exchange: An overview. In: International Conference on Provable Security, Springer (2014) 1–9
12. Druml, N., Menghin, M., Kuleta, A., Steger, C., Weiss, R., Bock, H., Haid, J.: A flexible and lightweight ecc-based authentication solution for resource constrained systems. In: Digital System Design (DSD), 2014 17th Euromicro Conference on, IEEE (2014) 372–378
13. Ulz, T., Pieber, T., Steger, C., Haas, S., Bock, H., Matischek, R.: Bring your own key for the industrial internet of things. In: Industrial Technology (ICIT), 2017 IEEE International Conference on, IEEE (2017) 1430–1435
14. Google: Google Weave - uWeave. https://weave.googlesource.com/weave/libuweave/+/HEAD (2016)