Cryptographic Protocols for Confidentiality, Authenticity and Privacy on Constrained Devices

1st Jan Hajny
Dept. of Telecommunications
Brno University of Technology
Brno, Czech Republic
hajny@feec.vutbr.cz

2nd Petr Dzurenda
Dept. of Telecommunications
Brno University of Technology
Brno, Czech Republic
dzurenda@feec.vutbr.cz

3rd Raul Casanova Marques
Dept. of Telecommunications
Brno University of Technology
Brno, Czech Republic
casanova@feec.vutbr.cz

4th Lukas Malina
Dept. of Telecommunications
Brno University of Technology
Brno, Czech Republic
malina@feec.vutbr.cz

***Abstract*—Cyber security and privacy protection play a crucial role in modern communication systems. While it is relatively easy to secure classical networks, it is a hard problem to provide even basic security properties, such as confidentiality, integrity, authenticity and privacy, in heterogeneous networks that involve devices with restricted resources. In these environments, such as industrial networks, sensor networks or IoT networks, the protection of user data is still very low. In this paper, we present the design of cryptographic protocols that provide the crucial security and privacy-protection features while they're fully implementable on constrained devices. First, we present a computationally efficient scheme for the establishment of a secure channel on a device with almost no cryptographic support and very low computational and memory resources. Second, we present a privacy-enhancing scheme for achieving so-called anonymous authentication, that is the verification of user authorization without disclosing her identity. Also in this case we use only very limited support of cryptographic operations and computational resources. Besides the full cryptographic description, we also show the benchmarks based on our implementation of protocols and the way of integration into real-world applications.**

***Index Terms*—authenticity, confidentiality, integrity, privacy, constrained devices**

I. Introduction

Security and privacy in communication networks is usually provided by cryptographic means. In case of security, we try to achieve confidentiality (a feature that guarantees that attackers cannot read data), integrity (a feature that guarantees that attackers cannot modify data during a transfer) and authenticity (a feature that guarantees that attackers cannot impersonate valid users). There are many proven ciphers that provide these CIA (Confidentiality, Integrity and Authenticity) features individually and even together in a single algorithm. The most popular standardized algorithms are Advanced Encryption Standard (AES) [1] for encryption (providing confidentiality), Secure Hash Function (SHA) [2], [3] for computing message digests (providing integrity) and (Elliptic Curve)Digital Signature Algorithm (EC)DSA [4] for digital signatures (providing authenticity).

The examples of algorithms mentioned above are integrated into cryptographic protocols that provide security features to users in communication systems. Most of the algorithms may be used in standard communication systems that involve servers, computers, mobile phones, etc. However, in heterogeneous networks, such as IoT and SCADA, there are also different types of devices, usually with very restricted computational and memory resources. As examples, we name sensors, controllers, wearable devices, cyber-physical devices or medical aids. On such devices with only very limited computational power, it is very difficult to use advanced cryptographic algorithms without affecting the fundamental functions. In most cases, the use of asymmetric ciphers (such as RSA, DSA or DH algorithms) is not supported by the CPUs and micro controllers and is very difficult to implement in software due to lack of computational power.

The situation gets even worse in case we need to implement privacy-preserving features on constrained devices. Even the fundamental building block for privacy-enhanced authentication, the group signatures [5]–[7], is based on asymmetric cryptography and operations that are at least as complex as digital signatures. More advanced cryptographic schemes, such as Attribute-Based Credentials (ABCs) [8]–[10] or Homomorphic Encryption (HE) [11]–[13] are currently not implementable on constrained devices due to lack of computational resources and unsupported basic operations, such as the bilinear pairings and arithmetic operations on elliptic curves.

However, with the increasing popularity of IoT and industrial networks, we cannot ignore security and privacy protection in real applications any more. Therefore, we propose practical cryptographic protocols for constrained devices that provide the basic security and privacy-protection features and are implementable on current hardware with reasonable running times.

A. State of the Art

The algorithms for regular hardware devices (i.e., computers, server, mobile devices) are well-known and standardized, usually by the National Institute of Standards and Technology (NIST): RSA or (EC)DSA for digital signatures, AES for encryption, SHA-2 or SHA-3 for message digests. These algorithms are used in cryptographic protocols for building secure channels, in most cases Transport Layer Security (TLS)

This paper is supported by the Czech Ministry of Industry and Trade grant #FV40340 and EU grant #813278 (A-WEAR).

[14] or IP Security (IPsec) [15] protocols are used. However, these protocols require asymmetric cryptography to authenticate users, which results in a very difficult or impossible implementation on constrained devices. Solutions based on symmetric cryptography exist, but are often proprietary (e.g., in case of smart-card based authentication systems [16]), too complex (e.g., Kerberos for enterprise networks [17]) or insecure (e.g., 2G mobile network authentication [18]). The performance of basic cryptographic pritmives has been evaluated on real-world constrained devices, e.g. in papers [19], [20].

In the area of privacy protection, we distinguish two major topics: anonymous routing and privacy-enhancing authentication systems. The anonymous routing protocols, such as TOR or Mixnets, are the prerequisite. Without anonymous routing, any higher-layer protection would be useless as users could be identified by IP addresses (or patterns in communication, content, etc.). Fortunately, there are already solutions [21], [22] that can be used even on constrained resources and achieve anonymous communication channels. However, the privacy-enhancing technologies for managing user identity and/or personal attributes, are not yet available on platforms with limited resources. There are very promising results from the smart card area [10], [23], [24], but generally, full-fledged user-centric authentication schemes are not yet ready for the implementation on existing constrained devices, in particular devices with restricted cryptographic support and strong requirements on low verification times. This paper provides solutions for applications witch such specific requirements.

B. Our Contribution

We propose two novel cryptographic schemes that provide 1) confidentiality, integrity and authenticity in networks involving constrained devices and 2) privacy protection for users using constrained devices and authentication devices. We fully describe the communication patterns, specify the cryptographic design and show the implementation results including benchmarks on ARM-based micro controllers and smart cards.

We provide the cryptographic preliminaries in Section II, the design of cryptographic protocols in Section III and the results from our practical implementation in Section IV.

II. Preliminaries

In this section, we describe the cryptographic preliminaries.

A. Notation

The symbol ":" means "such that" and $|x|$ is the bitlength of x. The symbol $\mathcal{H}$ denotes a secure hash function. We write $a \xleftarrow{\$} A$ when a is sampled uniformly at random from A. Let $\texttt{GroupSetup}(1^\kappa)$ be an efficient algorithm that generates a group $\mathbb{G} = \langle g \rangle$ of prime order q, such that $|q| = \kappa$. Let $\mathbf{e}$ denote a bilinear map.

B. Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order q. A bilinear map $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ must satisfy bilinearity, i.e., $\mathbf{e}(g_1^x, g_2^y) = \mathbf{e}(g_1, g_2)^{xy}$ for all $x, y \in \mathbb{Z}_q$; non-degeneracy, i.e., for all generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $\mathbf{e}(g_1, g_2)$ generates $\mathbb{G}_T$; and efficiency, i.e., there exists an efficient algorithm $\mathcal{G}(1^\tau)$ that outputs the bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$.

C. Weak Boneh-Boyen Signature

We recall the weak Boneh-Boyen signature scheme [6], which is existentially unforgeable against a weak (non-adaptive) chosen message attack under the q-SDH assumption.

$\texttt{Setup}$: On input security parameter τ, generate a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \leftarrow \mathcal{G}(1^\tau)$. Take $x \xleftarrow{\$} \mathbb{Z}_q$, compute $w = g_2^x$, and output $sk = x$ as private key and $pk = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathbf{e}, w)$ as public key.

$\texttt{Sign}$: On input message $m \in \mathbb{Z}_q$ and secret key sk, output $\sigma = g_1^{\frac{1}{x+m}}$.

$\texttt{Verify}$: On input the signature σ, message m, and public key pk, output 1 iff $\mathbf{e}(\sigma, w) \cdot \mathbf{e}(\sigma^m, g_2) = \mathbf{e}(g_1, g_2)$ holds.

III. Cryptographic Design

In this section, we specify the cryptographic protocols for security and privacy protection.

A. Security Protocols

The main goal of the security scheme is creating a security channel that provides confidentiality, integrity and authenticity. Therefore, there must be an authentication phase where at least users prove their identity. Based on the authentication phase, a shared key must be established and used for authenticated encryption. All these features must be realized using only symmetric cryptography (preferably only using AES and a hash function) so that it is implementable on existing, low-price microcontrollers, such as the PIC24 family from Microchip.

The communication pattern of our scheme employs the following entities:

- **Issuer** (I): is responsible for issuing personal identifiers ID_U and private keys K_{V-U} to users.
- **User** (U): gets the identifier ID_U and private key K_{V-U} and proves their possession to the verifier. Establishes a shared session key K_S for encryption.
- **Verifier** (V): verifies the possession of a secret keys K_{V-U}. Establishes a shared session key K_S for encryption.

Our scheme consists of the following algorithms:
$K \leftarrow \texttt{Setup}(1^\kappa)$: the algorithm inputs the security parameter κ and generates the master key $K \xleftarrow{\$} \mathbb{Z}_{2^\kappa}$. The $\texttt{Setup}$ algorithm is run by the issuer.

$(K_{V_i}, K_{V_i-U_i}) \leftarrow$`Issue`$(1^\kappa, K)$: the algorithm inputs the security parameter κ and the master key K. First, a randomly generated identifier ID_{V_i} is assigned and a unique secret key is generated for all verifiers using a key-derivation function (i.e., a hash-based PBKDF) as $K_{V_i} = \mathcal{H}(K, ID_{V_i})$. Then, a secret key unique per each verifier is generated to all users as $K_{V_i-U_i} = \mathcal{H}(K_{V_i}, ID_{U_i})$. The `Issue` algorithm is run by the issuer. Verifiers' and users' secret key are securely distributed to their owners.

$(K_S) \leftarrow$`Show`$(1^\kappa, ID_{U_i}, K_{V_i-U_i}, vnonce) \leftrightarrow$ `Verify`$(1^\kappa, ID_{V_i}, K_{V_i}, unonce) \rightarrow (K_S)$: the protocol consists of the `Show` algorithm run by the user and the `Verify` algorithm run by the verifier. The `Show` algorithm inputs the user's identifier, secret key and a random nonce from the verifier and outputs the session key K_S. The `Verify` algorithm inputs the verifier's identifier, secret key and the random nonce from the user and outputs the session key K_S. The protocol is a simple challenge-response protocol based on hashing secret keys with random challenges (nonces). The protocol is depicted in Figure 1.

Security Analysis
The protocol is based on a single primitive, namely the proven block cipher AES. The protocol provides the following features:

- **Confidentiality**: all communication data are sent inside the tunnel encrypted by AES.
- **Integrity**: data integrity is protected by the GCM mode of AES.
- **Authenticity**: both user and verifier are verified using secret keys.
- **Replay Attack Protection**: all authentication sessions are unique, randomized by nonces. Encryption key is always randomly generated.
- **Key Diversification**: all users and all verifiers have unique keys, so that the attacker breaking key of one user/verifier cannot get the keys of others.

B. Privacy Protocols

The design of security protocols is relatively easier than the construction of privacy-enhancing protocols, as only symmetric primitives are usually required. In this section, we design a protocol for so-called anonymous authentication. Using this protocol, the user can prove that she has the authorization to use a service without disclosing her identity. Therefore, all the transaction are anonymous, unlinkable and untraceable. In this scenario, we assume that issuer is also the verifier of the user authorisation (i.e., the private user's key ownership). Even in this settings, the verifier is unable to identify and/or trace a user to whom he already issued the keys. The protocol works with so-called epochs, therefore the keys must be re-issued periodically (e.g., each week or month).

The communication pattern employs the following entities:

- **Issuer** (I): is responsible for issuing personal identifiers ID_{Ui} and private keys (K_{Ui}, K'_{Ui}) to users.
- **User** (U): gets the identifier ID_{Ui} and private keys (K_{Ui}, K'_{Ui}) and proves their possession to the verifier. The proof is anonymous and always unique (randomized) so tracing and profiling of users is prevented.
- **Verifier** (V): verifies the possession of a secret key K_V. Using K_V, the verifier is able to check the user's proof of user key ownership without identifying the user or disclosing any keys. Verifier is also able to check that the key is valid for current time period (epoch).

Our scheme consists of the following algorithms:
$K \leftarrow$`Setup`(1^κ): the algorithm inputs the security parameter κ and generates the group $\mathbb{G} = \langle g_1 \rangle$ of prime order q, such that $|q| = \kappa$. The algorithm also generates the master key $K = \{K_0, K_1, K_2\} \xleftarrow{\$} \mathbb{Z}_q$. The `Setup` algorithm is run by the issuer.

$(K_{U_i}, K'_{U_i}, K_V) \leftarrow$`Issue`$(1^\kappa, K)$: the algorithm inputs the security parameter κ and the master key K. First, a randomly generated identifier ID_{Ui} and a current epoch identifier E (e.g., a week or month identifier) are set. Then, a unique secret key $K_{Ui} = g^{\frac{1}{K_0+K_1 ID_{Ui}+K_2 E}}$[1] and an auxiliary value $K'_{Ui} = K_{Ui}^{K_1}$ is generated for all users. The `Issue` algorithm is run by the issuer. Verifiers' and users' secret key are securely distributed to their owners.

$(\hat{K_{Ui}}, e, s, s_{ID}) \leftarrow$`Show`$(1^\kappa, ID_{U_i}, K_{Ui}, K'_{Ui}, nonce) \leftrightarrow$ `Verify`$(1^\kappa, K_V, \hat{K_{Ui}}, e, s, s_{ID}) \rightarrow (0/1)$: the protocol consists of the `Show` algorithm run by the user and the `Verify` algorithm run by the verifier. The `Show` algorithm inputs the user's identifier, secret key and the random nonce from the verifier and outputs the cryptographic proof of key ownership consisting of $(\hat{K_{Ui}}, e, s, s_{ID})$ values. The proof is based on the non-interactive proof of weak Boneh-Boyen signature knowledge (more details in [25]). The `Verify` algorithm inputs the verifier's secret key and the proof from the user and outputs 1 if the proof is correct and 0 otherwise. Note that neither user identifier not her keys are disclosed during the ownership proof. Furthermore, all communicated values are randomized in the sessions so that sessions of a single user cannot be linked together. The protocol is depicted in Figure 2.

Security Analysis
The protocol is using only a hash function and basic arithmetic in a standard prime-order group. The protocol provides the following features:

- **Anonymity**: users remain anonymous during the protocol execution.
- **Unlinkability**: all sessions of a single users cannot be linked together, so profiling is prevented.

[1]E denotes the identifier of current epoch, i.e. typically month, week or year.

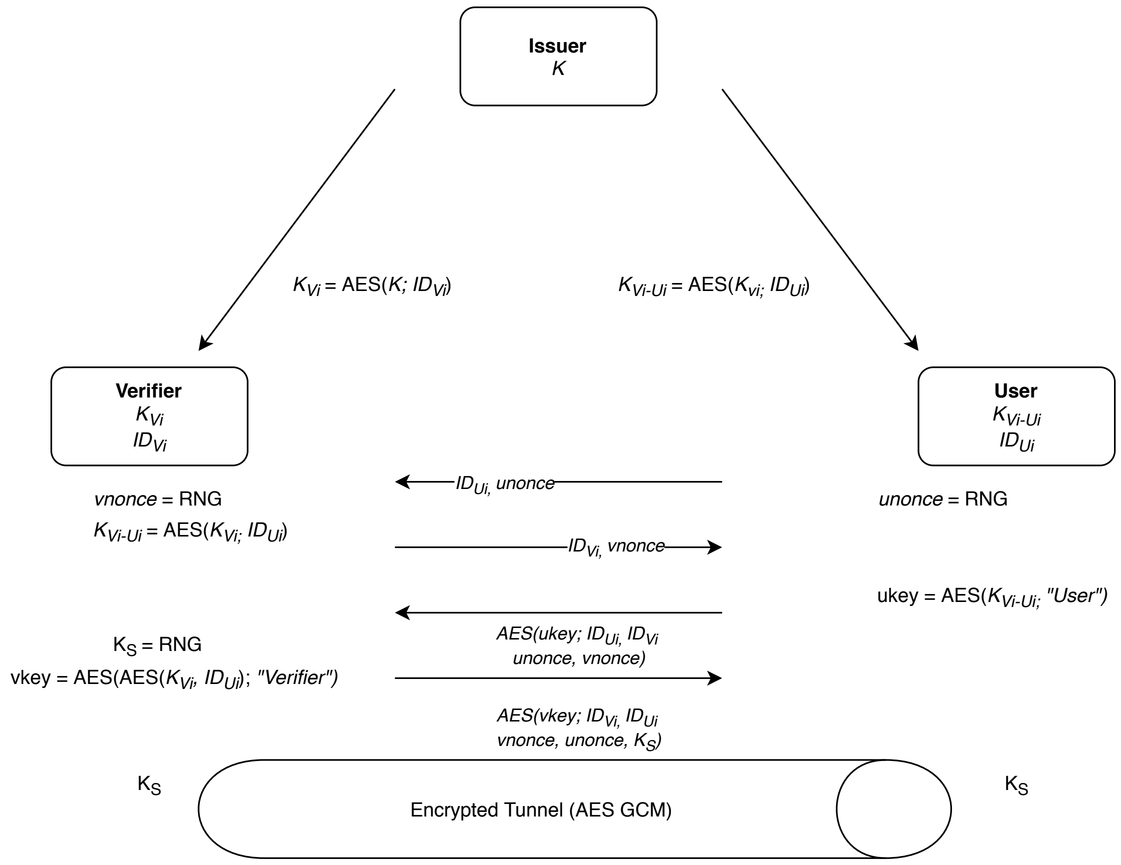
Issuer
K
K_Vi = AES(K; ID_Vi)
K_Vi-Ui = AES(K_vi; ID_Ui)
Verifier
K_Vi
ID_Vi
User
K_Vi-Ui
ID_Ui
ID_Ui, unonce
vnonce = RNG
K_Vi-Ui = AES(K_Vi; ID_Ui)
unonce = RNG
ID_Vi, vnonce
ukey = AES(K_Vi-Ui; "User")
AES(ukey; ID_Ui, ID_Vi unonce, vnonce)
K_S = RNG
vkey = AES(AES(K_Vi, ID_Ui); "Verifier")
AES(vkey; ID_Vi, ID_Ui vnonce, unonce, K_S)
K_S
Encrypted Tunnel (AES GCM)
K_S

Fig. 1. `Show` and `Verify` protocols of our security scheme.

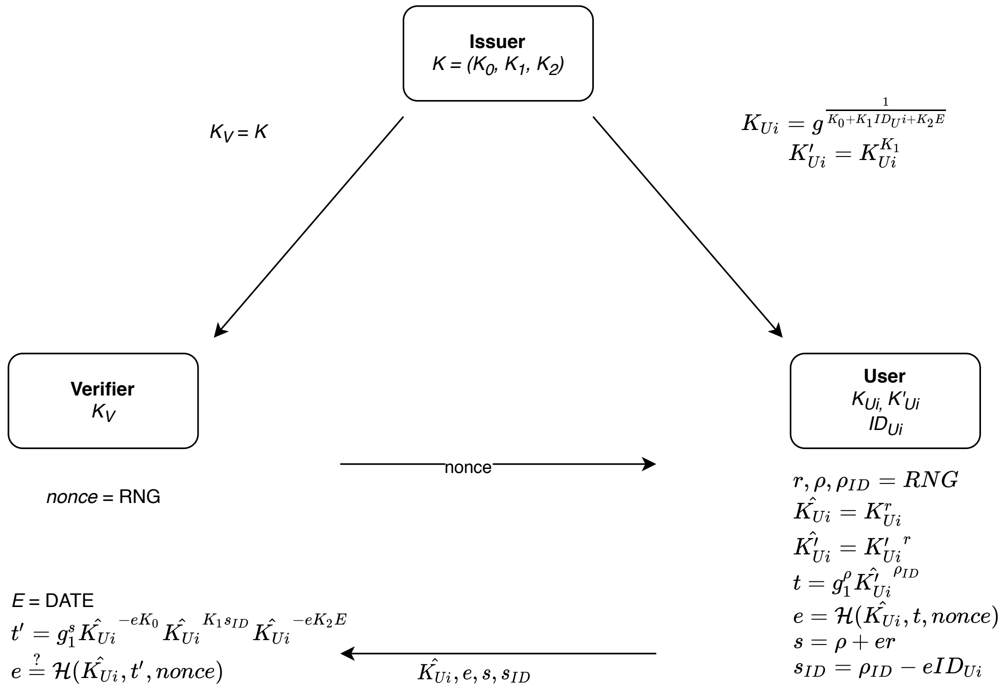
Issuer
K = (K_0, K_1, K_2)
K_V = K
$K_{Ui} = g^{\frac{1}{K_0+K_1 ID_U i+K_2 E}}$
$K'_{Ui} = K_{Ui}^{K_1}$
Verifier
K_V
User
K_Ui, K'_Ui
ID_Ui
nonce
nonce = RNG
$r, \rho, \rho_{ID} = RNG$
$\hat{K_{Ui}} = K_{Ui}^r$
$\hat{K'_{Ui}} = {K'_{Ui}}^r$
$t = g_1^{\rho} \hat{K'_{Ui}}^{\rho_{ID}}$
$e = \mathcal{H}(\hat{K_{Ui}}, t, nonce)$
$s = \rho + er$
$s_{ID} = \rho_{ID} - eID_{Ui}$
E = DATE
$t' = g_1^s \hat{K_{Ui}}^{-eK_0} \hat{K_{Ui}}^{K_1 s_{ID}} \hat{K_{Ui}}^{-eK_2 E}$
$e \overset{?}{=} \mathcal{H}(\hat{K_{Ui}}, t', nonce)$
$\hat{K_{Ui}}, e, s, s_{ID}$

Fig. 2. `Show` and `Verify` protocols of our privacy-preserving scheme.

- **Untraceability**: although issuer issues keys to users, he cannot link the issuance and verification sessions. This feature holds also in cases where issuer is also verifier.
- **Revocation**: users' keys expire after the epoch period (e.g., a prepaid time) is over. Users have to ask for a key regeneration, but that can be rejected by issuers if users should be revoked from the system.

IV. Implementation Results

We provide the proof-of-concept implementation of both designed schemes, see Figure 1 and Figure 2 for more details. Both schemes' implementations consist of the terminal side, i.e., entity representing the issuer/verifier, and the authentication token side, i.e., entity representing the user. All implementations are available on the GitHub public repository: https://github.com/a-wear/crypto-cap.git. The terminal application is written in C programming language, and includes only publicly available libraries: OpenSSL [26] (performing encryption and hashing algorithms), MCL [27] (performing operations over elliptic curves, such as EC scalar point multiplication and EC point addition), GMP [28] (performing modular arithmetic operations) and PCSC [29] (providing communication with a smart card). The terminal application was always executed on a Raspberry Pi 4B device.

In case of the authentication token, we explore different devices such as smart cards and smartphones, which are widely used in physical access control systems. We used only publicly available software libraries, development tools and off-the-shelf devices. Namely, MultOS API, SmartDeck 3.0.1 and MUtil 2.8 are used for the development of a MultOS application, Oracle Java Card API version 3.0.4 is used to develop a Java Card application, and finally, Android API (Java programming language) version 4.4 (KitKat) with BouncyCastle [30] library (performing elliptic curve operations) is used to a develop mobile application. The technical and software specification of all deployed devices is provided in Table I.

TABLE I
Technical and software specification of deployed devices.

Device	Type	OS Version
Java Card J3H145	`Smart Card`	`JavaCard 3.0.4`
MultOS ML4	`Smart Card`	`MultOS 4.3.1`
Samsung Galaxy S5	`Smart Phone`	`Android 6.0.1`
Raspberry Pi 4B	`Microcomputer`	`Raspberry Pi OS`

In our implementation, we use only standard off-the-shelf programmable smart cards, namely MultOS ML4 contact smart card (MCU SC23Z018, 1.75 kB RAM, 252 kB ROM, 18 kB EEPROM, OS MultOSv4.3.1) with support of modular arithmetic and elliptic curve operations (EC scalar point multiplication and EC point addition), see [31] for more details. Therefore, MultOS platform seems to be the best option for a Protocol 2 implementation that is depicted in Figure 2. Unfortunately, the MultOS ML4 card implementation lacks T=1 transmission protocol, therefore, only data of maximum 255 bytelength can by transmitted in one APDU (Application Data Unit) message. These cards are also missing contactless communication interface and many of modern cryptographic algorithms, e.g., AES, block cipher modes, e.g., GCM, or secure hash functions, e.g., SHA-2. On the other hand, Java Card supports elliptic curve cryptography (EC scalar point multiplication and EC point addition from Java Card API version 3.0.5). However, those cards are missing modular arithmetic operations and modern block cipher modes. Indeed, Java Card 3.0.5 includes `javacardx.framework.math.BigNumber`, i.e., a support of big integer operations, beyond modular arithmetic operations, and `javacardx.crypto.AEADCipher`, i.e., a support of AES-GCM and AES-CCM algorithms. However, these classes are only optional and they are not usually supported by off-the-shelf smart cards. Therefore, due to the lack of cryptographic support, we were able to implement the Protocol 1 on Java Cards and the Protocol 2 on MultOS cards only. However, both protocols have been implemented on an Android device. By employing smart phones with OS Android and using the BouncyCastle library, we can achieve higher performance and use more secure cryptographic algorithms in accordance with the Protocol 1 and Protocol 2 design. Table II presents used security parameters, key sizes and cryptographic algorithms in our implementation.

TABLE II
Security parameters, key sizes and cryptographic algorithms.

	Java Card	**MultOS**	**Samsung Galaxy S5**
Protocol 1 (security scheme), see Figure 1			
K, K_{Vi}, K_{Vi-Ui}	16 B	N/A	16 B
Ks, $vkey$, $ukey$	16 B	N/A	16 B
ID_{Vi}, ID_{Ui}	16 B	N/A	16 B
$vnonce$, $unonce$	16 B	N/A	16 B
IV	16 B	N/A	12 B
Key derivation function	AES-ECB	N/A	AES-ECB
	NoPadding	N/A	NoPadding
Encrypted tunnel	AES-CBC	N/A	AES-GCM
	NoPadding	N/A	NoPadding
Protocol 2 (privacy-preserving scheme), see Figure 2			
K_0, K_1, K_2	N/A	32 B	32 B
K_{Ui}, K'_{Ui}	N/A	65 B	65 B
ID_{Ui}	N/A	32 B	32 B
$nonce$, r, ρ, ρ_{ID}	N/A	32 B	32 B
DATE	N/A	4 B	4 B
Hash algorithm	N/A	SHA-1	SHA-1
Eliptic curve	N/A	BN-256	BN-256

The user-side algorithm execution is the most crucial part of the authentication protocol, see Figure 3. In fact, the time needed to execute a verification algorithm on Raspberry Pi 4B takes only 0.2 ms for Protocol 1 and 5.3 ms for Protocol 2. In both cases, the Android phone shows better performance results comparing with the smart cards (Java Card and MultOS card). However, it is at the expense of security, since smartphones are not considered as tamper-resistant devices in contrast to smart cards.

V. Conclusion

In this paper, we proposed two cryptographic protocols for constrained devices. The Security Protocol is designed

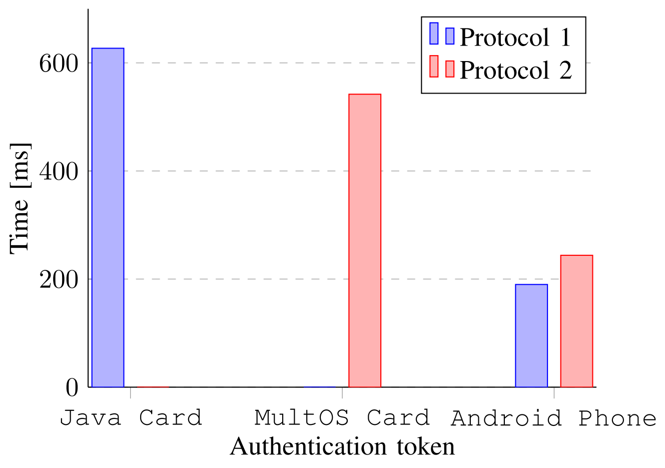

Fig. 3. Speed comparison of our `Show` algorithms on various devices.

for devices which have almost no cryptographic support and provide only random number generation and AES encryption. Using the Security Protocol, a secure communication channel with both confidentiality and integrity protection can be established. The Privacy Protocol requires only random number generation and standard arithmetic operations over EC. Using the Privacy Protocol, users' registration can be verified without disclosing her identity. Furthermore, standard privacy-enhancing features are provided: unlinkability and untraceability of transactions. Besides the theoretical design, we also present the performance measurement results using the implementation of both protocols on smart cards and mobile devices. The benchmarks indicate that both protocols can be run in less than 0,7 s on any constrained device, including smart cards. As the next steps, we plan to further optimize the protocols and implement on wearable devices, such as smart watch.

Acknowledgment

This paper is supported by the Czech Ministry of Industry and Trade grant # FV40340. The authors gratefully acknowledge funding from European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska Curie grant agreement No. 813278 (A-WEAR: A network for dynamic wearable applications with privacy constraints, http://www.a-wear.eu/).

References

[1] J. Daemen and V. Rijmen, "Reijndael: The advanced encryption standard." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 26, no. 3, pp. 137–139, 2001.
[2] W. Penard and T. van Werkhoven, "On the secure hash algorithm family," *Cryptography in Context*, pp. 1–18, 2008.
[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The keccak sha-3 submission," *Submission to NIST (Round 3)*, vol. 6, no. 7, p. 16, 2011.
[4] T. Pornin, "Deterministic usage of the digital signature algorithm (dsa) and elliptic curve digital signature algorithm (ecdsa)," *Internet Engineering Task Force RFC*, vol. 6979, pp. 1–79, 2013.
[5] J. Camenisch and M. Stadler, *Efficient group signature schemes for large groups*, 1997, pp. 410–424.
[6] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *Journal of Cryptology*, pp. 149–177, 2008.
[7] S. Ling, K. Nguyen, H. Wang, and Y. Xu, "Constant-size group signatures from lattices," in *IACR International Workshop on Public Key Cryptography*. Springer, 2018, pp. 58–88.
[8] J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. Ø. Pedersen, "Scientific comparison of ABC protocols," 2014.
[9] J. Hajny and L. Malina, *Unlinkable Attribute-Based Credentials with Practical Revocation on Smart-Cards*, 2013, pp. 62–76.
[10] J. Camenisch, M. Drijvers, and J. Hajny, "Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs," in *ACM CCS WPES'16 Proceedings*, 2016, pp. 123–133.
[11] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford university Stanford, 2009, vol. 20, no. 9.
[12] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.
[13] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.
[14] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," 2008.
[15] S. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W. Ritchey, and S. R. Sharma, "Guide to ipsec vpns," *NIST Special Publication*, vol. 800, pp. 800–77, 2005.
[16] G. de Koning Gans, J.-H. Hoepman, and F. D. Garcia, "A practical attack on the mifare classic," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2008, pp. 267–282.
[17] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33–38, 1994.
[18] D. Abodunrin, Y. Miche, and S. Holtmanns, "Some dangers from 2g networks legacy support and a possible mitigation," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 585–593.
[19] J. Hajny, L. Malina, Z. Martinasek, and O. Tethal, "Performance evaluation of primitives for privacy-enhancing cryptography on current smart-cards and smart-phones," in *ESORICS DPM*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 17–33.
[20] A. Ometov, P. Masek, L. Malina, R. Florea, J. Hosek, S. Andreev, J. Hajny, J. Niutanen, and Y. Koucheryavy, "Feasibility characterization of cryptographic primitives for constrained (wearable) iot devices," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016, pp. 1–6.
[21] D. Chaum, D. Das, F. Javani, A. Kate, A. Krasnova, J. De Ruiter, and A. T. Sherman, "cmix: Mixing with minimal real-time asymmetric cryptographic operations," in *International Conference on Applied Cryptography and Network Security*. Springer, 2017, pp. 557–578.
[22] F. W. Baumann, U. Odefey, S. Hudert, M. Falkenthal, and U. Breitenbücher, "Utilising the tor network for iot addressing and connectivity." in *CLOSER*, 2018, pp. 27–34.
[23] A. de la Piedra, J.-H. Hoepman, and P. Vullers, *Towards a Full-Featured Implementation of Attribute Based Credentials on Smart Cards*, 2014.
[24] J. Camenisch, M. Drijvers, P. Dzurenda, and J. Hajny, "Fast keyed-verification anonymous credentials on standard smart cards," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2019, pp. 286–298.
[25] D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups," *Journal of cryptology*, vol. 21, no. 2, pp. 149–177, 2008.
[26] https://www.openssl, "Openssl: The open source toolkit for ssl/tls," org/docs/manmaster/crypto/crypto.htm, 2014.
[27] M. Shigeo, "Mcl library," https://github.com/herumi/mcl, 2018.
[28] T. Granlund and the GMP development team, "The gnu mp bignum library," https://gmplib.org/, 2016.
[29] D. Corcoran and L. Rousseau, "Pcsc lite project: Middleware to access a smart card using scard api (pc/sc)," https://pcsclite.apdu.fr/, 2018.
[30] L. of the Bouncy Castle Inc., "The legion of the bouncy castle," https://www.bouncycastle.org/, 2013.
[31] P. Dzurenda, S. Ricci, J. Hajny, and L. Malina, "Performance analysis and comparison of different elliptic curves on smart cards," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2017, pp. 365–36 509.