Privacy ABCs: Now Ready for Your Wallets!

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

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

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

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

***Abstract*—The paper deals with privacy-enhanced electronic access control technologies, in particular cryptographic schemes that allow verification of users' personal attributes without their identification, so-called anonymous attribute-based credential schemes (ABCs). We present the last bit necessary for making ABCs practical for large-scale applications that are using smart cards as users' devices for storing credentials: a novel cryptographic scheme that combines fast credential verification protocols with efficient offline revocation protocols. Using proven building blocks, namely weak Boneh-Boyen (wBB) signatures, keyed-verification credentials and k-times anonymous proofs, we construct a practical scheme for proving personal attributes anonymously, unlinkably, untraceably and, most importantly, with the verifier-local revocation (VLR) functionality that is running on standard existing smart cards. To prove the practicality of the design, we implemented all the proposed protocols using an off-the-shelf card, benchmarked the proving protocol, compared to existing solutions and put all the source codes on the GitHub as an open source. The cryptographic design and our implementation are efficient enough to be immediately used for the privacy enhancement of existing large-scale applications, such as electronic ID cards (e-IDs), public transportation cards, apps for citizen tracing during pandemic situations or secure authentication of IoT devices.**

***Index Terms*—Access control, anonymity, identity, privacy, smart cards.**

I. Introduction

Using anonymous attribute-based credentials (ABCs), users can prove their personal attributes (such as age, citizenship, ticket ownership, negative SARS-CoV-2 test) without revealing their identity. Furthermore, advanced privacy-enhancing features are provided by ABC schemes, such as unlinkability, untraceability or selective attribute disclosure.

While the ABC schemes are known for a long time since the publication of [1]–[3], their implementation on constrained offline devices was a hard problem for a long time due to high computational complexity. In particular, the implementations of core protocols on smart cards were impractical until very recently [4]–[7]. Implementations with efficient large-scale revocation are still completely missing on smart cards and only available on online and computationally strong user devices, which usually do not include IoT devices, which have only very limited memory and computational resources.

In this paper, we finally present a scheme that holds all privacy-enhancing features, is provably secure, provides efficient revocation even in applications with millions of users and yet it is implemented and benchmarked on a standard smart card. Since we consider smart cards the most convenient devices for storing and proving personal attributes due to their security, durability and portability, we believe that the results presented in this paper will contribute to the practical deployment of ABC technologies in applications such as eID cards, e-ticketing, mass transportation or privacy-enhanced people tracing and smart quarantine.

A. State of the Art

There are several implementations of ABC schemes on programmable smart cards available, such as [4], [5], [7], [8]. However, these implementations lack revocation, which is the crucial feature for removing misbehaving or invalid users from the system. Revocation was the topic of many papers [9]–[14], but none of them proposed practical protocols that can be used in large-scale applications with smart cards due to the following issues: use of unsupported operations (e.g., bilinear pairing on constrained user's device), need for periodic updates of a smart card content, need for online communication, loss of unlinkability, only user-driven revocation or missing security proofs. Lueks *et al.* [15] proposed a revocation scheme with low computational cost based on the Vuller's and Alpar's (VA's) Idemix implementation [4] that is part of the IRMA Project. The disadvantages are limited unlinkability within one epoch and need for revocation list re-computation for each verifier, which is impractical. The scheme was further extended by Verheul [16] to avoid the disadvantages but requires bilinear pairings, that are currently unsupported by smart cards. Efficient revocation scheme for smart-cards was proposed by Camenisch et al. in [6]. However, the integration of the revocation protocols with any ABC scheme is not described, nor implemented yet. Recently, Camenisch et al. [7] present Keyed-Verification Anonymous attribute-based Credentials (KVAC) based on algebraic MAC and Boneh-Boyen signatures. The solution is designed directly for smart cards. However, the scheme lacks revocation completely.

B. Our Contribution

ABC schemes without practical revocation are useless for electronic service providers, as they essentially need mechanisms for the identification and revocation of misbehaving users. However, the cryptographic protocols for proving personal attributes and for the revocation of invalid users have been handled rather separately so far. While there exist schemes that allow efficient proving of attributes using smart cards (e.g., [7]) and there are generic schemes for ABC

revocation (e.g., [6]), there is also a lack of results showing at least the basic principles of their combination. Due to the high complexity of proving and revocation protocols, it is a non-trivial task to design mechanisms for their integration, to combine these protocols together and keep the computational complexity within the limits of existing smart cards. Therefore, the first contribution of this paper is the full cryptographic specification of a novel scheme that combines efficient attribute proving protocols with revocation protocols.

Although there exist theoretical specifications of some partial ABC solutions, the practical implementations are currently only very few [4], [5]. That is caused by the complexity of the development on the smart card platforms. The lack of computational and memory resources, very limited cryptographic support, awkward debugging and missing arithmetic operations (such as operations on elliptic curves, bilinear pairings, etc.) result in the absence of even prototypical implementations of ABCs that could be used by service providers for their testing and integration into practical systems. To improve this situation, we provide the full open-source implementation[1] of all protocols of our revocable ABC scheme that is fully functional on off-the-shelf smartcard with a standard operating system. This is the second major contribution of the paper.

Finally, due to the use of advanced cryptographic primitives in ABC protocols and due to very limited resources of smart cards, the existing partial implementations are often very inefficient. It is not uncommon that some protocols take over 10 seconds [17], [18], which is useless in real applications such as public transportation, loyalty cards or ticketing. As a third major contribution, we present benchmarks of our protocols on real devices which prove the efficiency our cryptographic design and implementation and give estimates of expected running times to future adopters of the ABC technology.

II. Cryptographic Design

A. Preliminaries

The symbol ":" means "such that", the symbol "||" means concatenation 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 $\mathbf{e}$ denote a bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

B. Protocol Description

The communication pattern depicted in Fig. 1 employs the following entities:

- **Revocation Authority** (RA): assigns and issues a unique revocation handler (private attribute m_r) to each user using the `Issue` algorithm. Thanks to this attribute, the revocation authority can revoke users.
- **Issuer** (I): is responsible for issuing attributes (personal attributes m_i, such as gender, nationality, age or negative SARS-CoV-2 test) to a user aggregated in a cryptographic credential *cred* using the `Issue` algorithm. The credential is digitally signed by the issuer's secret key sk_I.
- **User** (U): gets the credential *cred* which includes issued attributes from the issuer and anonymously proves the possession of attributes to the verifier using the `Show` algorithm. Furthermore, the user has to compute a one-time per-session pseudonym C which is linked to the credential *cred* via the revocation handler m_r and prove that this one-time pseudonym is not revoked.
- **Verifier** (V): verifies the possession of required attributes and the revocation status of the revocation handler using the `Verify` algorithm and Verifier's secret key sk_V and RA's public key pk_{RA}.

The algorithms and protocols are specified in the next section, including their input and output parameters:

- $(sk_I, sk_V, params_I) \leftarrow$`SetupI`$(1^\kappa)$: the algorithm inputs the security parameter κ and generates the public system parameters (which are implicit inputs of all other algorithms), namely a bilinear group with parameters $params_I = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, \mathbf{e})$ satisfying $|q| = \kappa$. Next, the private keys $sk_I = sk_V = (x_0, \ldots, x_{n-1}, x_r) \xleftarrow{\$} {\mathbb{Z}_q}^2$, where n is the number of all attributes in the credential, are generated and securely distributed to issuer and verifier. In our implementation, we use a bilinear group generated by the MCL [19] library, namely we use the BN-254 curve. The `SetupI` algorithm is run by the issuer.

- $(params_{RA}, sk_{RA}, pk_{RA}, RL, RH) \leftarrow$`SetupRA` $(1^\kappa, ver_{max})$: the algorithm inputs the security parameter κ and the parameter ver_{max} setting the maximum number of unlinkable sessions per user within one epoch. First, the RA computes its keypair as $(sk_{RA} \xleftarrow{\$} \mathbb{Z}_q, pk_{RA} = g_2^{sk_{RA}})$. Second, the algorithm sets integers $(k, j) : ver_{max} = k^j$. In our implementation, we set $j = 2, k = 10$, i.e. 100 pseudonyms per epoch (e.g., a day). Furthermore, the algorithm chooses random integers $(\alpha_1, \ldots, \alpha_j) \xleftarrow{\$} \mathbb{Z}_q$ and computes $h_z = g_1^{\alpha_z}$ for all z from 1 to j. Finally, the RA generates an empty revocation list RL and an empty list of revocation handlers RH. The algorithm outputs keys (sk_{RA}, pk_{RA}) and parameters $params_{RA} = (k, j, (h_1, \ldots, h_j), (\alpha_1, \ldots, \alpha_j))$. The `SetupRA` algorithm is run by the revocation authority, pk_{RA} and $params_{RA}$ are securely distributed to other entities.

- $(\sigma, \sigma_{x_1}, \ldots, \sigma_{x_{n-1}}, \sigma_{x_r}, m_r, \{(e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})\}, RH) \leftarrow$`Issue` $(sk_{RA}, sk_I, RH, (m_1, \ldots, m_{n-1}), ID)$: the algorithm inputs the private key of the revocation authority sk_{RA}, the issuer's private key sk_I, a list of personal attributes $(m_1, \ldots, m_{n-1})$, user's ID and the list of all revocation handlers of all users RH. The

[1] GitHub link: https://github.com/rcasanovama/rkvac-protocol-multos

[2] Similarly to [7], we assume merging Issuer and Verifier roles, but this is not required by the cryptographic design, as x_0 may remain private to Issuers only and avoided in the `Verify` algorithm.

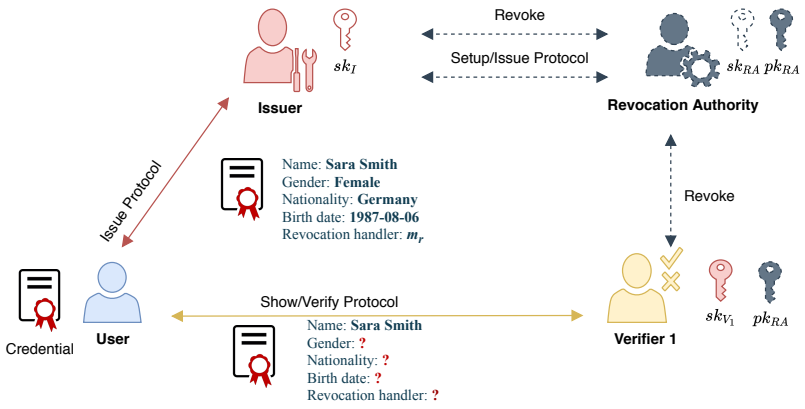

Fig. 1. Block diagram of our revocable ABC scheme.

algorithm outputs signature on all user attributes σ (i.e., cryptographic credential), the revocation handler m_r, the (signed) randomizers and updates the list of revocation handlers RH. The algorithm consists of two sub-algorithms: `IssueRA` and `IssueI`. The `IssueRA` algorithm is run first and after that it is followed by the `IssueI` algorithm.

- The `IssueRA`: the algorithm is run between the user and the revocation authority. RA chooses randomizers $(e_1, \ldots, e_k) \xleftarrow{\$} \mathbb{Z}_q$ and signs each of them using the wBB signature [20]: i.e. $\sigma_{e_z} \leftarrow g_1^{\frac{1}{e_z + sk_{RA}}}$ for all z from 1 to k. RA also chooses a random and unique revocation handler m_r and signs it together with the user's identifier ID, i.e. computes $\sigma_{RA} = g_1^{\frac{1}{\mathcal{H}(m_r||ID)+sk_{RA}}}$. RA updates its list of revocation handlers $RH = RH + (m_r||ID + \{(e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})\})$ and sends (m_r, σ_{RA}) and pairs $\{(e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})\}$ back to the user.
- The `IssueI`: the algorithm is run between the user and the issuer. The user sends all its attributes $(m_1, \ldots, m_{n-1}, m_r)$ and RA's signature σ_{RA} to the issuer. The issuer checks the signature validity, signs all required attributes with the issuer's secret key as $\sigma = g_1^{\frac{1}{x_0 + m_1 x_1 + \cdots + m_{n-1} x_{n-1} + m_r x_r}}$ and computes auxiliary values $\sigma_{x_i} = \sigma^{x_i}$ for $1 \leq i \leq n$. The algorithm outputs cryptographic credential σ and auxiliary values $(\sigma_{x_1}, \ldots, \sigma_{x_{n-1}}, \sigma_{x_r})$ to the user.

- $(C, \pi) \leftarrow$`Show`$((m_1, \ldots, m_{n-1}, m_r), (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_{n-1}}, \sigma_{x_r}), \{(e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})\}, m_{z \in D}, epoch) \leftrightarrow$ `Verify`$(sk_V, pk_{RA}, m_{z \in D}, C, \pi, RL, epoch) \rightarrow (0/1)$: on the user's side, the algorithm inputs user's attributes $(m_1, \ldots, m_{n-1}, m_r)$, the signature σ (i.e. cryptographic credential), the set of randomization pairs $\{(e_1, \sigma_{e_1}), \ldots, (e_k, \sigma_{e_k})\}$, the indices of disclosed attributes $m_{z \in D}$ and the identifier of the current time epoch $epoch$. On the verifier side, the algorithm inputs the verifier's private key sk_V, the revocation lists RL and the epoch identifier. The user outputs the pseudonym C and the cryptographic proof π of the attributes possession. The verifier outputs logical value 0/1, i.e. permit or deny access. The algorithms `Show` and `Verify` are run between the user and the verifier. The detailed description of the `Show` and `Verify` algorithms is depicted in Figure 2. First, the user computes one-time pseudonym C by hashing the epoch identifier $epoch$, a unique per-session value $i = \sum_{z=1}^{j} \alpha_z e_z$, where e_z and α_z are secret user's parameters (stored on a secure device such as a smart card) and the revocation handler m_r. The user then randomizes its credential and computes a proof of knowledge of all its attributes inside the credential. Furthermore, the user proves that the one-time pseudonym C and signature $\hat{\sigma}$ are constructed using the same handler m_r. Finally, the verifier verifies the proof π and checks whether the one-time pseudonym C is not placed on the revocation list RL.

- $RL' \leftarrow$`Revoke`$(RH, RL, sk_{RA}, \pi, C)$: the algorithm inputs the RA's private list of revocation handlers RH, the public revocation list RL, the RA's private key and the communication transcript π, C received from the verifier. The algorithm outputs updated revocation list RL'. The `Revoke` algorithm is typically run between the verifier and the revocation authority. The RA is able to reconstruct all pseudonyms of all users for every epoch by computing $C = g_1^{\frac{1}{i - m_r + \mathcal{H}(epoch)}}$, where $i = \alpha_1 e_I + \alpha_2 e_{II}$ is computed for all possible combinations of α_j and e_k (i.e. in case of $j = 2, k = 10$, it is 100 combinations) and m_r, which are taken from the list of revocation handlers RH. By comparing the C received from the verifier with all reconstructed pseudonyms of all users, the RA is able to identify the owner of the pseudonym (since RA stores the link between the revocation handler m_r and the identity of the user in RH) and put all his one-time pseudonyms on the updated revocation list RL for the next epoch.

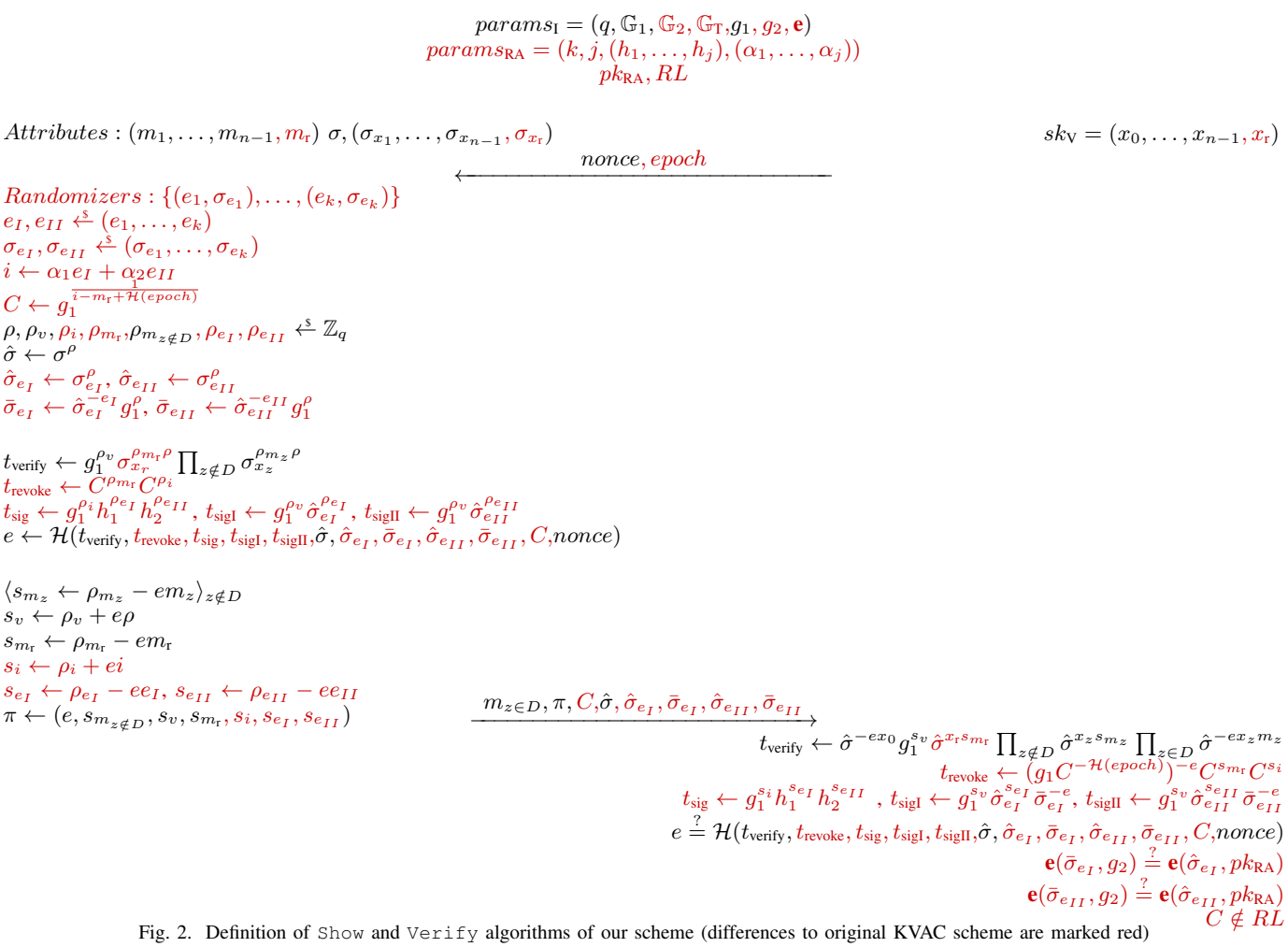

Fig. 2. Definition of `Show` and `Verify` algorithms of our scheme (differences to original KVAC scheme are marked red)

III. Security Analysis

Our scheme is composed of two crucial components: the keyed-verification credential scheme [7] and the scalable revocation scheme [6]. In the next sections, we outline the security model and the results of formal analysis. More details, including the formal security models and formal security proofs, can be found in our aforementioned papers on these building blocks.

A. Keyed-Verification Credentials

We follow the security model defined by Chase et al. [21], thus requiring (informally):

- *Correctness*: honest users are almost always accepted,
- *Unforgeability*: dishonest users are almost always rejected,
- *Anonymity*: protocol transcripts are indistinguishable from simulations,
- *Key-Parameter Consistency*: there is a unique private key to every public key/parameter.

The keyed-verification credential scheme has been proven secure in the random oracle model in [7] under the n-SCDHI assumption (defined below).

Definition 1 (n-Strong Computational Diffie-Hellman Inversion Problem (SCDHI)): Let $\mathcal{O}^{\mathsf{bb}}(\cdot)$ on input $(m_1, \ldots, m_n) \in \mathbb{Z}_q^{*n}$ add $(m_1, \ldots, m_n)$ to Q and output $g^{1/(x_0 + \sum_{i=1}^{n} x_i m_i)}$. Let $\mathcal{O}^{\mathsf{dh}_i}(\cdot)$ on input h output h^{x_i}. Define the advantage of $\mathcal{A}$ as follows.

$$\mathsf{Adv}_{n-SCDHI}(\mathcal{A}) = \mathsf{Pr}\Big[(\mathbb{G}, g, q) \leftarrow \mathsf{GroupSetup}(1^{\kappa}), (x_0, \ldots, x_n) \xleftarrow{\$} \mathbb{Z}_q^{*n+1}, (y, m_1^*, \ldots, m_n^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{bb}}(\cdot), \mathcal{O}^{\mathsf{dh}_0}(\cdot), \ldots, \mathcal{O}^{\mathsf{dh}_n}(\cdot)}(g) : y = g^{\frac{1}{x_0 + \sum_{i=1}^{n} x_i m_i^*}} \wedge (m_1^*, \ldots, m_n^* \notin Q\Big].$$

SCDHI is (t, ϵ)-hard if no t-time adversary has advantage at least ϵ.

B. Scalable Revocation Scheme

We use the security model defined in [6] with following requirements (informally):

- *Revocation Completeness*: unrevoked users will almost always pass the revocation check.
- *Revocation Soundness*: revoked users will almost always be rejected by the revocation check.
- *Revocation Privacy*: protocols will not release any private data during proving the possession of an unrevoked revocation handler.

The revocation scheme has been proven secure in the random oracle model in [6] under the l-DBDHI assumption [22] (defined below).

Definition 2 (l-Decisional Bilinear Diffie-Hellman Inversion Problem (DBDHI)):

Input the tuple $(g, g^x, \ldots, g^{x^q})$. Distinguish $\mathbf{e}(g,g)^{1/x}$ from random γ in $\mathbb{G}$ with advantage:

$$\mathsf{Adv}_{l-DBDHI}(\mathcal{A}) = |\mathsf{Pr}[\mathcal{A}(g, g^x, \ldots, g^{x^q}, \mathbf{e}(g,g)^{1/x}) = 1] - \mathsf{Pr}[\mathcal{A}(g, g^x, \ldots, g^{x^q}, \gamma) = 1]| \leq \epsilon$$

DBDHI is (t, ϵ)-hard if no t-time adversary has advantage at least ϵ.

IV. Implementation Results

We provide the proof-of-concept implementation of our scheme in order to benchmark and compare it with the existing schemes. The scheme implementation consists of the smart card side (i.e. entity representing the user) and the terminal side (i.e. entity representing the issuer/verifier and the revocation authority). Both implementations are available on the GitHub public repository: Link Anonymized. In case of the smart card application, only standard MultOS API and free public development environment (Eclipse IDE for C/C++ Developers, SmartDeck 3.0.1, MUtil 2.8) were used. We used standard off-the-shelf programmable smart cards, namely MultOS ML4 contact smart cards (MCU SC23Z018, 1.75 kB RAM, 252 kB ROM, 18 kB EEPROM, OS MultOSv4.3.1), which support only T=0 transmission protocol. The MultOS platform was selected due to its wide support of modular arithmetic and elliptic curve operations (ECC scalar multiplication and ECC addition), see [23] for more details. For the terminal application, OpenSSL [24], MCL [19] and GMP [25] libraries were used. The GMP library achieved the best performance results from all tested libraries which supports bilinear pairing operations, see [26] for more details.

Figure 3 depicts the comparison between our implementation (blue and red) and the original keyed-verification scheme lacking revocation [7] (black and white) for different numbers of attributes stored and disclosed. The figure shows the times for maximum $n-1$ disclosed attributes, where n is the number of stored attributes on the card since the revocation handler (the attribute m_r) is always present but never disclosed. The `Show` and `Verify` algorithms of both schemes were implemented using a pairing-friendly BN-254 curve generated by the MCL library. We stress that the original keyed-verification implementation had to be extended to support BN-254 curve since it supported only the NIST P-192 curve. The algorithm time (in red) shows the time necessary to compute all operations on the card. The overhead time (in blue) includes the additional time for APDU transmission and MultOS processing. Results are the arithmetic means of 10 measurements in milliseconds.

Independently of the number of attributes used, our scheme requires always only ca. 1.3 s to generate the ownership proof with all but revocation attributes disclosed. The total time of around 2.08 s is necessary for the proof generation including communication (2.01 s) and computations on a terminal (0.07 s on Raspberry Pi 4 Model B, ARM Cortex-A72, 4 GB RAM, Raspberry Pi OS 4.19 32b). The complexity of the scheme increases with the number of undisclosed attributes, each undisclosed attribute increases the `Show` time by ca. 100 ms. Our implementation is limited to 10 attributes per user, but the available memory resources (approx. 1.75 KB RAM and 7.5 KB usable EEPROM) would allow storing up to 50 attributes on a card.

V. Conclusion

We focused on the practical aspects of ABC technologies and presented an integrated scheme that holds all privacy-preserving features, provides efficient revocation and is based on provably secure building blocks. The scheme has been implemented on a standard smart card, benchmarked and the source is openly available on GitHub. To the best of our knowledge, we present the first implementation of a revocable anonymous credential scheme on a standard smart-card that achieves practical running times. Next, we plan to further optimize the `Show` protocol, keep searching for faster cards and start the real-world pilot deployment.

Acknowledgment

This paper is supported by the Technology Agency of the Czech Republic grant # TL02000398: Legal and technical tools for privacy protection in cyberspace. 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/) and EC H2020 project SPARTA, Grant No. 830892.

References

[1] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, Oct. 1985. [Online]. Available: http://doi.acm.org/10.1145/4372.4373

[2] J. Camenisch and A. Lysyanskaya, *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 93–118.

[3] S. A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. Cambridge, MA, USA: MIT Press, 2000.

[4] P. Vullers and G. Alpar, "Efficient selective disclosure on smart cards using idemix," in *Policies and Research in Identity Management*. Springer, 2013, vol. 396, pp. 53–67.

[5] W. Mostowski and P. Vullers, "Efficient U-Prove implementation for anonymous credentials on smart cards," in *International Conference on Security and Privacy in Communication Systems*. Berlin, Heidelberg: Springer, 2011, pp. 243–260.

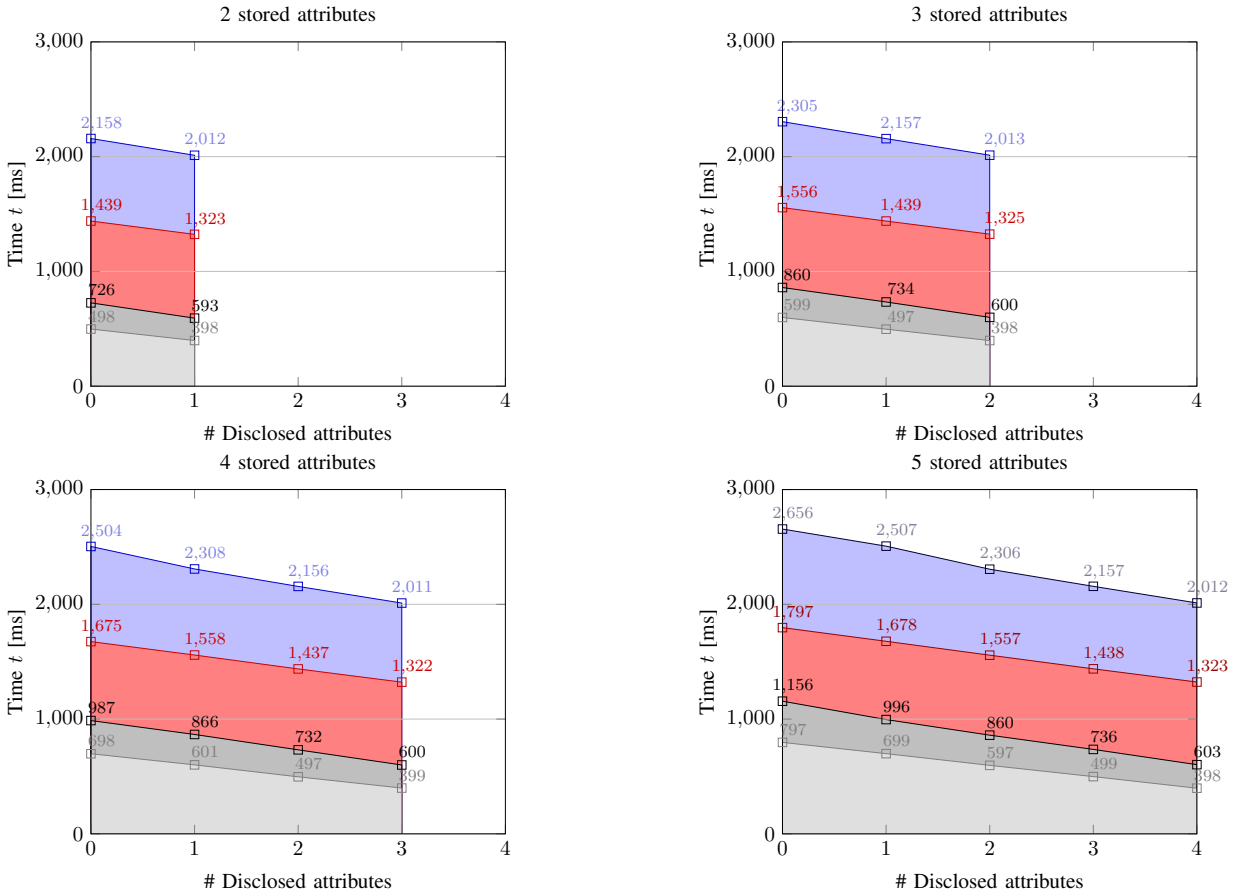

Fig. 3. Speed comparison of our `Show` algorithm implementation with the original keyed-verification implementation lacking revocation mechanisms [7]. Red - our algorithm time, blue - our total time with overhead, light grey - original algorithm time and dark grey - original total time with overhead.

[6] J. Camenisch, M. Drijvers, and J. Hajny, "Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs," ser. WPES '16. New York, NY, USA: ACM, 2016, pp. 123–133. [Online]. Available: http://doi.acm.org/10.1145/2994620.2994625

[7] 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.

[8] A. De La Piedra, J.-H. Hoepman, and P. Vullers, "Towards a full-featured implementation of attribute based credentials on smart cards," in *International Conference on Cryptology and Network Security*. Springer, 2014, pp. 270–289.

[9] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, "Blacklistable anonymous credentials: Blocking misbehaving users without ttps," in *Proceedings of the 14th ACM CCS*, ser. CCS 07. New York, NY, USA: ACM, 2007, p. 7281.

[10] J. Camenisch, M. Kohlweiss, and C. Soriente, "Solving revocation with efficient update of anonymous credentials," in *Security and Cryptography for Networks*. Springer, 2010, pp. 454–471.

[11] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 61–76.

[12] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Topics in Cryptology – CT-RSA 2005*, A. Menezes, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 275–292.

[13] J. Hajny, P. Dzurenda, and L. Malina, "Privacy-pac: Privacy-enhanced physical access control," 11 2014, pp. 93–96.

[14] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith, "Nymble: Blocking misbehaving users in anonymizing networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 256–269, 2011.

[15] W. Lueks, G. Alpár, J.-H. Hoepman, and P. Vullers, "Fast revocation of attribute-based credentials for both users and verifiers," *Computers & Security*, vol. 67, pp. 308–323, 2017.

[16] E. R. Verheul, "Practical backward unlinkable revocation in fido, german e-id, idemix and u-prove," Cryptology ePrint Archive, Report 2016/217, 2016, https://eprint.iacr.org/2016/217.

[17] H. Tews and B. Jacobs, "Performance issues of selective disclosure and blinded issuing protocols on java card," in *Information Security Theory and Practice*. Berlin, Heidelberg: Springer, 2009, pp. 95–111.

[18] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup, "Anonymous credentials on a standard java card," in *Proceedings of the 16th ACM conference on Computer and communications security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 600–610.

[19] M. Shigeo, "Mcl library," https://github.com/herumi/mcl, 2018.

[20] 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.

[21] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic MACs and keyed-verification anonymous credentials," in *CCS '14*. New York, NY, USA: ACM, 2014, pp. 1205–1216. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660328

[22] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Public Key Cryptography - PKC 2005*, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 416–431.

[23] P. Dzurenda, S. Ricci, J. Hajny, and L. Malina, "Performance analysis and comparison of different elliptic curves on smart cards," in *International Conference on Privacy, Security and Trust (PST)*, 2017, pp. 1–10, calgary, Canada, ISBN: 978-1-5386-2487-6.

[24] "Openssl: The open source toolkit for ssl/tls," 2014. [Online]. Available: https://www.openssl.org/docs/manmaster/crypto/crypto.html

[25] "The gnu mp bignum library," 2014. [Online]. Available: https://gmplib.org/

[26] J. Hajny, P. Dzurenda, S. Ricci, L. Malina, and K. Vrba, "Performance analysis of pairing-based elliptic curve cryptography on constrained devices," in *ICUMT'18*. IEEE, 2018, pp. 1–5.