Dedicated Hardware for Attribute-Based Credential Verification

Geoffrey Ottoy, Jorn Lapon, Vincent Naessens, Bart Preneel, Lieven Strycker

Dedicated Hardware for Attribute-Based Credential Verification

Geoffrey Ottoy[1], Jorn Lapon[2], Vincent Naessens[2]
Bart Preneel[3], and Lieven De Strycker[1]

[1] KAHO Sint-Lieven, DraMCo research group,
Gebroeders de Smetstraat 1, 9000 Gent, Belgium
geoffrey.ottoy@kahosl.be
http://www.dramco.be/
[2] KAHO Sint-Lieven, MSEC research group,
Gebroeders de Smetstraat 1, 9000 Gent, Belgium
jorn.lapon@kahosl.be http://www.msec.be
[3] KU Leuven, COSIC and IBT,
Kasteelpark Arenberg 10, bus 2446, 3001 Leuven-Heverlee, Belgium
http://www.esat.kuleuven.be/cosic/

Abstract. Attribute-based credentials systems offer a privacy-friendly solution to access electronic services. In this field, most research has been directed into optimizing the prover operations and exploring the usability boundaries on mobile platforms like smart cards and mobile phones. This research assumes that the verification of credential proofs occur at a powerful back end. However, a broad range of (embedded) applications lack this powerful back end.
This article shows that hardware accelerators for modular exponentiations, greatly reduce the run time of applications that require credential verification in an embedded context. In addition, when verification requires a considerable amount of the total run time (i.e., communication included), the use of dual-base (simultaneous) exponentiation hardware further increases the overall performance.
All tests have been performed in a practical setup between a smartphone and an embedded terminal using NFC communication.

Key words: attribute-based credentials, dedicated hardware, embedded, NFC

1 Introduction

Today, a lot of personal information is revealed through the use of electronic services. Although service providers require only little information for running their service, often many more attributes are collected. For instance, when a user needs to prove that he is older than 18 to access a gambling site, when using an electronic identity card based on certificate technology, other personal data are disclosed as well. Moreover, all actions of the same user can be linked. Attribute-based credentials [12, 10, 11, 8] offer a more privacy-friendly solution

to access electronic services. First, they support selective disclosure. This means that a user can opt to reveal only a subset of possible properties of the attributes embedded in the credential. For instance, a credential with the user's date of birth as an attribute, can be used to prove that the owner is over 18, without disclosing the exact date of birth or other attributes. Second, some attribute-based credential systems also support accountability and/or unlinkability of transactions. The former allows to identify individuals conditionally (e.g., in case of abusive behavior). The latter means that multiple actions of the same user cannot be linked.

Two major classes of attribute-based credential systems exist. The first class [8] uses *blind signatures* [12] in order to break the link between the issuer and the user's credential. In short, the issuer signs the user's credential, without knowing the resulting signature value. As a result, when the credential is used, even if the issuer and relying party share information, it cannot be linked to the issuance phase. In order to make multiple transactions unlinkable, a batch of credentials can be issued and for each anonymous transaction, a new credential will be used. The second class, also called CL-based credentials [10, 11], uses *zero-knowledge proofs* to break the link between the credential issuance and its use. During authentication, the user proves in zero-knowledge to the relying party, that she has a genuine credential (i.e., certified by the trusted issuer). Here, zero-knowledge means that, in the general case, nothing is disclosed except the fact that the credential is genuine, and the prover is the holder of the credential (i.e., she knows the corresponding private key). An implementation is available for each class: `U-Prove` [32], implements the first class, and `Identity Mixer` [17], belongs to the second class.

A major disadvantage of attribute-based credential technologies is their poor performance – both in terms of processing power and memory footprint – compared to certificate technology. For instance, in case of `Identity Mixer`, zero-knowledge proofs require multiple exponentiations by the prover and the verifier. This currently leads to unacceptable response times in many application domains. Existing research focuses on optimizing the prover operations and exploring the usability boundaries on mobile platforms like smart cards and mobile phones. This research assumes that the verification of credential proofs occurs at a powerful back end. However, selective disclosure is also very relevant in settings where a user authenticates to a terminal that is not connected with a powerful back end. For instance, a stand-alone cigarette or beverage vending machine wants to verify if the user is older than 18. Similarly, a waste disposal center wants to restrict access to local inhabitants. Therefore, users need to prove to live in a certain city before they are granted access to that location. In both situations, users do not want to release too much information to discourage extensive profiling.

This paper especially targets hardware support to accelerate the *verification* of credential attributes released during authentication. We, thereby, focus on CL-based credentials. The main contributions are threefold. First, a hardware platform is presented that supports the acceleration of a wide range of crypto-

graphic operations. Second, the feasibility of the platform is demonstrated by integrating it in an access control terminal with NFC communication capabilities. The terminal is applied in a scenario where a user stores an attribute-based credential on her NFC-enabled smartphone, and selectively discloses some attributes to get access. Third, the terminal is extensively evaluated. More specifically, the performance is compared to a terminal without dedicated hardware support. The effect of the number of attributes and whether they are revealed, is studied as well.

The rest of this paper is structured as follows. Section 2 and 3 respectively point to related work and offer background information. Thereafter, the case study is presented in Section 4. Section 5 focuses and the realization of the prototype. Next, the hardware platform is evaluated in Section 6. We end up with conclusions in Section 7.

2 Related work

Attribute-based credentials based on CL signatures, require substantially more computational effort, for both proving and verifying, than traditional authentication technologies. This is mainly due to a substantial number of exponentiations during zero-knowledge proofs. Hence, performance might be a bottleneck, especially if credentials are stored on mobile carriers (like smart cards or mobile devices), and proofs are generated on those carriers. In literature, multiple implementations of attribute-based credentials in such environments have been presented. Bichsel et al. [4] presented a full Java Card implementation of attribute-based credentials. Computing the credential proof took about 7.4 s for a 1280-bit modulus, and up to 16.5 s for a 1984-bit modulus. In [3] and [33], to increase efficiency, the computation of the proof is divided between a tamper-proof smartcard and a partially trusted host. This approach, however, requires partial trust on the host. Recently, Vullers and Alpár [38] also implemented the CL-based prover protocol on a MULTOS card. The computation of the proof only takes about 1.1s (for a 1024-bit modulus) for a simple credential proof. This is an interesting result that shows that also CL-based credentials on smartcards may become practical in the near future.

In contrast to the CL-based schemes, there are also prototypes implementing `U-Prove` [32] attribute-based credentials, which take about 5 s for showing a credential [37]. Later, Mostowski and Vullers [20] implemented the same protocol on a MULTOS [15] card with better support for modular arithmetic, resulting in only about 0.5 s. Note that in order to preserve unlinkability, the U-Prove system requires the issuance of a new credential for each transaction, which may quickly exhaust the EEPROM of the card [4].

Unfortunately, all of these solutions evaluate and/or optimize the performance of the prover side. Our work, on the other hand, mainly focuses on the fast verification of attribute-based credentials in resource-constrained environments. It is clear that faster – and simultaneous – calculation of modular exponentiations may reduce the response times at the verifier, and hence, increase

the overall performance of an attribute-based authentication procedure. Many hardware implementations and optimizations are presented in the literature.

The most straightforward way of performing a modular exponentiation is by repeated squarings and multiplications to get the final result [6, 34]. The square and multiply step can be performed either in parallel [22] or sequentially [13]. Booth encoding is also used to increase the number of "00"-combinations of exponent bits in simultaneous exponentiation [18]. This reduces the number of multiplications and thus results in a speedup. However, more memory is required to store precomputed values. Also for simultaneous exponentiations, the square-and-multiply approach can be followed. A major drawback lies in the fact that the required amount of memory increases exponentially with the number of exponentiations carried out simultaneously.

As most (if not all) exponentiation algorithms rely on multiplications, it is not surprising that many implementations of hardware multipliers have been described in the literature. Almost all of these multipliers rely on Montgomery's algorithm [19]. This algorithm allows for very efficient hardware implementations. The algorithm's main disadvantage is the carry propagation. Several architectures have been proposed that combine Montgomery multiplication with either a redundant radix number system [31, 16] or the Residue Number System [2, 28] to cope with this problem. Unfortunately, these implementations have several other drawbacks. For instance, a lot of preprocessing of the operands is required.

The best known class of Montgomery multipliers is the systolic array architecture. Nedja and Mourelle [22] have shown that for operands larger than 512 bit, the systolic array implementation improves the $time \times area$ product over other implementations. They compared their work with that of Blum and Paar [5], which was one of the first milestones for systolic array implementations. Systolic array Montgomery multipliers have been implemented in 2-dimensional [22, 21] and 1-dimensional designs [24]. The 1-dimensional variant requires less silicon to implement, but is slower than the 2-dimensional implementation.

The k-partition method is a different way to speed up Montgomery multiplication [23]. In this method, k partitions operating in radix 2^k, each of which computes a part of the total result. The fastest multiplication would execute in n/k cycles. The complexity of the partitions, however, is higher than for a standard Montgomery multiplier, but often the FPGA's on-board multipliers are used to implement the partitions.

Montgomery multipliers can use Booth encoding [7] to replace the use of precomputed hard multiples in the processing elements. In [29], this is combined with left-shifting of the input operands –instead of right-shifting the result– to reduce the critical path. In [1, 30], more flexibility is added to the design by varying the length of the processing elements and by implementing the Booth encoding in hardware rather than requiring precomputed hard multiples of the input operands.

Tenca and Koç [35] came up with a scalable pipelined design where the circuit of the Montgomery multiplier is split into "word size" processing elements. Since then, several improvements have been made to their original design [36, 39].

3 Verification of CL based credentials proofs.

We briefly recall the protocol for the verification of a CL-based credential proof, which is part of the signature proof of knowledge. This is also the protocol that will be running on the terminal. For a full definition of the CL signature scheme, the signature proof of knowledge and the construction of the proof, we refer to Appendix A.

The verifier verifies the proof as follows:

1. Compute:

$$\hat{Z} = \left(\frac{Z}{\prod_{j \in A_r} R_j^{m_j} (A')^{2^{l_e - 1}}} \right)^{-c} (A')^{\hat{e}} \left(\prod_{i \in A_h} R_i^{\hat{m}_i} \right) (S^{\hat{v}'}) \mod n \qquad (1)$$

2. Verify results:

$$\hat{m}_i \stackrel{?}{\in} \{0,1\}^{l_m + l_\phi + l_H + 1} \quad \forall i \in A_h$$
$$\hat{e} \stackrel{?}{\in} \pm\{0,1\}^{l_{e'} + l_\phi l_H + 1}$$
$$c \stackrel{?}{=} H(context, A', \hat{Z}, n_1)$$

The proof is rejected if any of these checks fail.

4 Realization of the Embedded Terminal

4.1 Platform Specification

To analyze attribute-based credential verification on an embedded terminal, we use an FPGA-based test platform [25, 27]. This allows us to easily change the hardware configuration of the design. Furthermore, our platform implements a standard embedded design setup, so conclusions drawn with this platform can be generalized to non-FPGA-based systems.

Fig. 1 shows the structure of the embedded test terminal. It is based on a Xilinx ML605 evaluation board housing a Virtex 6 FPGA. The central controller is a MicroBlaze embedded processor running embedded Linux. Using Linux (in our case the PetaLinux distribution[4]) has the big advantage that standard libraries become available on the embedded system; specifically we use GMP[5] for the large number arithmetic and libnfc for the NFC communication[6].

An NXP PN532 development kit implements the RF front end for the NFC communication and is able to emulate different types of NFC devices like, Mifare, ISO14443-A/B, DEP, Felica, etc. We configure the PN532 as an NFC initiator scanning for ISO14443-A tags. A Google Nexus S smartphone running CyanogenMod 9.1 in card emulation mode acts as the prover.[7]

[4] Petalinux software development kit information: `http://www.xilinx.com/tools/petalinux-sdk.htm`

[5] GMP project website: `http://gmplib.org/`

[6] libnfc project website: `http://www.libnfc.org/`

[7] Cyanogenmod project website: `http://www.cyanogenmod.org/`

Hardware accelerators are provided for both the hashing (SHA-256) and for the exponentiations. Furthermore, the platform offers various types of volatile and non-volatile memory as well as several types of I/O (e.g., LEDs and switches) The complete platform, both CPU and hardware cores, run at a frequency of 100 MHz.

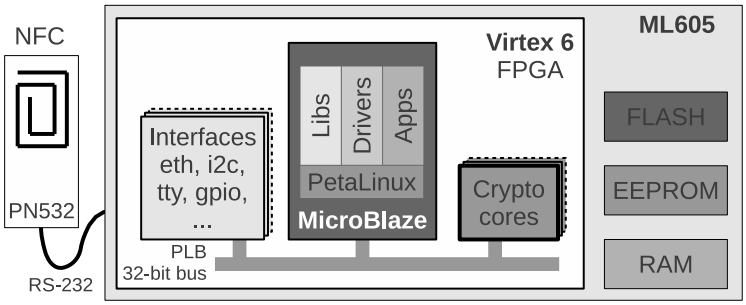


Fig. 1. Embedded test terminal setup

4.2 Crypto Core Specification

The hardware accelerator for the exponentiations [26] is specifically designed to carry out dual-base exponentiations (2) simultaneously, but it can also be used to compute a single-base exponentiation or a modular multiplication. The kernel of the accelerator is a pipelined 1-dimensional systolic array Montgomery multiplier. The design and hardware documentation can be found at: `http://opencores.org/projects/mod_sim_exp`.

$$g_0^{e_0} \cdot g_1^{e_1} \bmod m \tag{2}$$

The lengths of the exponents can be chosen freely by the controlling software. However, the construction of the hardware requires that both exponents are of the same length in case of dual-base exponentiations. This means that a shorter exponent needs to be padded with preceding zeros to match the length of the longer exponent. It also needs to be noted that all exponents are interpreted as positive integers. Modular inverses are computed in software.

Run times for multiplication and average run times for single-base exponentiation and dual-base exponentiation can be computed by equations (3), (4), (5) respectively.

$$t_{\text{mult.}} = \left\lceil \frac{n}{s} + 2 \cdot (n-1) \right\rceil / f_{clk} \tag{3}$$

$$t_{\text{1-exp.}} = \frac{3}{2} \cdot w_0 \cdot t_{\text{mult.}} \tag{4}$$

$$t_{\text{2-exp.}} = \left[\frac{3}{2} \cdot (w_0 - w_1) + \frac{7}{4} \cdot w_1 \right] \cdot t_{\text{mult.}} \tag{5}$$

With:

n the operand length [# bits]
s the pipeline stage length [# bits]
f_{clk} the clock frequency [Hz]
w_0, w_1 the length of the exponents e_0, e_1 where $w_1 \leq w_0$ [# bits]

Note that due to the bus traffic and latency introduced by the OS, an overhead of 5 to 6 ms needs to be taken into account for the exponentiations.

4.3 Implementation Details

To apply the hardware accelerator, the credential verification protocol (1) is rearranged and split into a modular product of dual-base exponentiations as presented in (6). The A' exponent $\left(c \cdot 2^{l_e - 1} + \hat{e}\right)$ as well as all the modular inverses are computed in software. All multiplications are computed in hardware.

$$\hat{Z} = \underbrace{\left(Z^{-1}\right)^{c} \cdot R_0^{\hat{m}_0}}_{\text{(a)}} \cdot \underbrace{\left(A'\right)^{\left(c \cdot 2^{l_e - 1} + \hat{e}\right)} \cdot S^{\hat{v}'}}_{\text{(b)}} \cdot \underbrace{\prod_{j \in A_r} \left(R_j^{-1}\right)^{m_j} \cdot \prod_{i \in A_h} R_i^{\hat{m}_i}}_{\text{attribute-dependent}} \mod n \tag{6}$$

Special attention is required when using the hardware to perform dual-base exponentiations. From equation (5), it follows that a minimal run time can be achieved when the length of the exponents differs as little as possible. That is why the verification step is rearranged and computed as shown in (6). For the attribute-based part, the revealed and hidden attributes are grouped in separate dual-base exponentiations where possible.

5 Tests and results

In prior work, only little timing results on the verification of attribute-based credentials are available and they are often hard to compare. Nevertheless, Table 1 presents a comparison of our embedded terminal with comparable implementations available in literature. A precise comparison is difficult because of the different architectures, processor speeds and implementations. Still, the figures clearly show the value of our hardware accelerated solution with respect to general purpose devices. Note that the measurements by Dietrich [14] are performed based on the DAA verification protocol, which is closely related to the CL credential verification used in this paper.

Table 1. Timing results (in ms) for the verification of attribute-based credentials, with only a master secret, compared to prior work.

n	Bichsel [3]	Dietrich [14](DAA)			This paper
	Intel Core 2-Duo	Intel Core 2-Duo	P910	Nokia 6131	
	P9600 @ 2.53GHz	T7500 @ 2.2GHz	ARM9 @ 156 MHz	ARM9 @ 229 MHz	@ 100MHz
1024	78	40	8240	16960	124
1536	187	-	-	-	215
2048	375	110	22100	64270	-

5.1 High level description of tests

As pointed out before, we are interested in the run time of the credential verification protocol on an embedded terminal. To that end, different cases are evaluated on the test platform described in Section 4. All tests are performed with a modulus length of both 1024-bit and 1536-bit. The hardware accelerator has 16-bit stages and operates at 100 MHz; this is also the clock frequency of the embedded CPU.

With these test cases, several questions are addressed:

- What is the effect of hardware acceleration on the run time? We will compare an embedded software implementation –using GMP on the MicroBlaze CPU– with an implementation using hardware offload, both using single-base exponentiations and dual-base exponentiations.
- What is the effect of the number of attributes in the credentials on the verification run time?
- What is the effect of the number of hidden/revealed attributes?
- To compare the verification run time with the total protocol run time, we will also take into account the communication overhead.

5.2 Verification Performance

As a first test, the verification of a credential with a single attribute (i.e., the master secret) is evaluated. Table 2 presents the run time of an embedded software implementation (*SW*), compared with an implementation using the hardware accelerator; both with single-base (*1-exp*) and dual-base exponentiations (*2-exp*). The table also shows the NFC communication time (i.e., the time required to transmit the proof π) as part of the overall protocol run time.

As expected, the verification with embedded software takes significantly longer than the hardware accelerated implementations. For a modulus of 1536 bits, it takes about 14 seconds or 95% of the total run time. The hardware accelerator clearly improves the performance of the verification and hence the overall run time. Moreover, the main share of the run time shifts towards the communication over NFC. The Table also shows that the communication speed is not constant, even for the same modulus lengths. Although the results are averages, this is

Table 2. Comparison of the average timing results for the credential proof verification on an embedded platform, where all computations are performed in software (*SW*), and where a hardware accelerator is used for single-base *1-exp* and dual-base *2-exp* exponentiations. The credential contains a single hidden attribute (m_0, the master secret).

		NFC Communication		**Verification**		**Hash and check**		**Total**
n	Implementation	[ms]	[%]	[ms]	[%]	[ms]	[%]	[ms]
1024	*SW*	748	12	5696	88	7	0	6451
	1-exp	733	82	159	18	7	1	899
	2-exp	721	85	124	15	7	1	852
1536	*SW*	765	5	13846	95	9	0	14620
	1-exp	782	76	240	23	9	1	1031
	2-exp	768	77	215	22	9	1	992

mainly due to overhead introduced in the operating system; keep in mind that Linux is not a real-time operating system.

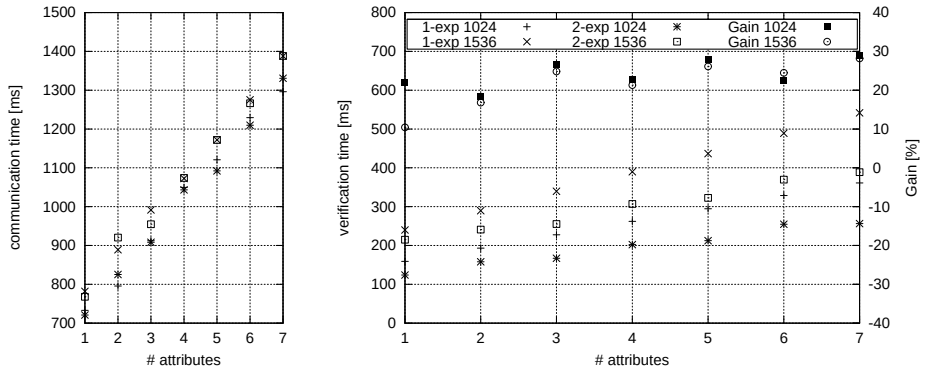
The tests also illustrate that a different size of the modulus has much more effect (relatively) on the verification time than on the communication time. This is the case for all three implementations. As could be expected, the time for hashing and making the necessary checks is negligible with respect to the time required for running the verification protocol.

5.3 Influence of the number of attributes

Second, the effect of the number of attributes in the credential is examined. Fig. 2 shows the run time for the communication (a) and the verification (b) with respect to the number of attributes in the credential. Fig. 2(b) also presents the gain in run time of dual-base exponentiations with respect to single-base exponentiations, defined as:

$$\text{Gain} = \frac{T_{\textit{1-exp}} - T_{\textit{2-exp}}}{T_{\textit{1-exp}}} \cdot 100\%$$

Increasing the number of attributes increases both communication time and verification time. This is obvious, as the proof π will also be larger. If a single-base exponentiation accelerator is used, the verification time increases linearly with the number of attributes. However, if dual-base exponentiations are used, the verification time increases stepwise. This is due to the fact that the time for computing a dual-base exponentiation is comparable to a single-base exponentiation. Hence, based on equation 6, when verifying a credential with an odd number of attributes, our dual-base exponentiations can be used to their full extent. In contrast, in the case of an even number of attributes, a single-base exponentiation is required. In other words, increasing the number of attributes from an even number to an odd number, results in replacing a single-base exponentiation by a dual-base exponentiation, which requires only a small overhead.



(a) Communication (b) Verification

Fig. 2. Influence of the number of attributes on the run time. All attributes remain hidden.

Increasing the number of attributes from an odd number to an even number results in having an additional single-base exponentiation.

As can be seen, verifying more attributes also increases the gain for using dual-base exponentiations instead of single-base exponentiations (e.g., about 30% gain for a credential with 7 attributes). The gain is roughly the same for both 1024-bit and 1536-bit moduli.

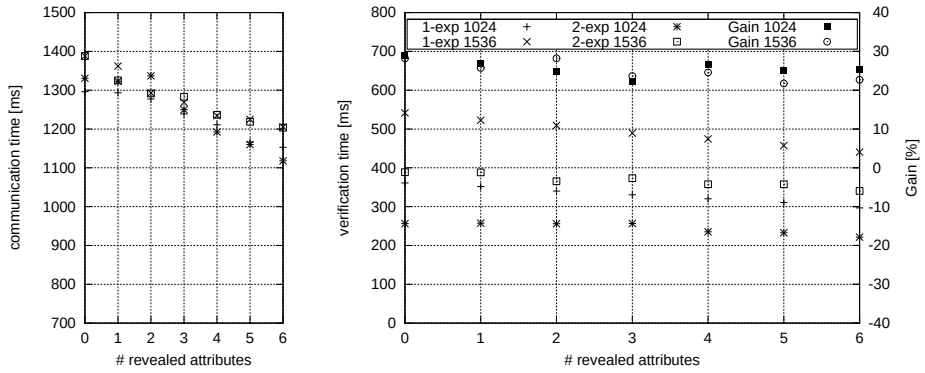
5.4 Influence of the number of revealed attributes

Finally, the influence of the number of revealed attributes on the run time has been examined. The number of revealed attributes varies from 0 to 6 (i.e., the master secret is never revealed). The communication time and verification run time is set out as well as the gain for using dual-base exponentiations (Fig. 3).

Both the communication time and verification time decrease with an increasing number of revealed attributes. This is because the exponent m_j of a revealed attribute is shorter than the exponent $\hat{m}_i$ of a hidden attribute. Again, there is a linear relationship when single exponentiations are used and a stepwise behavior for dual-base exponentiations. Note that for our tests the size of the attribute values was 256 bits, while in reality this will often be much smaller. In fact, an attribute representing, for instance, the user's gender, could require only a single bit. Hence, the decrease in time when releasing more attributes would become even more significant.

6 Evaluation

It is clear that for embedded applications the use of our hardware accelerator for credential verification greatly increases the feasibility in terms of run time.



(a) Communication (b) Verification

Fig. 3. Influence of the number of revealed attributes on the run time. The credential contains 7 attributes (master secret included).

Compared to a powerful back end as is used in [3, 14], the run times in this embedded context are of the same order of magnitude. Thus, applications such as physical access control (e.g., opening a door) or stand-alone vending machines that support privacy, anonymity and/or unlinkability become feasible.

Note that for testing purposes, the current hardware accelerator was implemented on FPGA. For several applications where form factor or energy consumption are key requirements, this is undesirable. However, the same accelerator can be implemented on ASIC (as a first step) or a custom IC (a final product), which are better suited for commercial applications. The advantages demonstrated with our prototype implementation remain valid in these settings as well.

The current hardware accelerator is not tamper-proof and resistant against side-channel attacks. For applications that only require credential verification, this is not problematic. If however, disclosed information must not be learned by other parties than the verifier, extra measures should be taken. For instance, to prevent timing analysis (which may reveal for instance if more or less attributes are disclosed), exponentiations should be performed with a constant timing; more specifically with the worst case timing: $(2 \cdot w_0 \cdot t_{\text{mult}})$. Obviously this means a decrease in performance, which is the same for both single-base and dual-base exponentiations. Hence, multi-base exponentiations get even more attractive in this setup.[8]

The scenario presented in this paper, in which NFC is used for communication, shows that the gain between single or dual-base exponentiations is only marginal with respect to the overhead caused by the communication. The figures in this paper where obtained using NFC @ 106 kbits. Clearly, faster communication (e.g., NFC @ 424 kbit/s or Bluetooth v2.1 @ +1 Mbit/s) makes the use of our dual-base exponentiation accelerator more interesting.

[8] Note that for an l-base exponentiation, 2^l n-bit memory locations are required.

The efficiency of the dual-base hardware accelerator is maximal in the case of exponents of the same length. However, in the prototype, the difference in the size of the exponents in dual-base exponentiation (b) of Eq. 6, is significant. A solution is to split exponentiation $S^{\hat{v}'}$ into a dual-base exponentiation $S_1^{\hat{v_1}'}.S_2^{\hat{v_2}'}$ with smaller exponents (see [9, 4] for more details). This, however, may require a slight modification of the prover protocol.

In this article we only examined the speedup of the verification of attribute-based credential proofs on embedded platforms. Note that for a complete setup, certificate revocation should also be supported.

7 Conclusions

In this paper, we present the use of a hardware accelerator for modular exponentiations, in order to reduce the run time of applications that require attribute-based credential verification in an embedded context. In addition, the use of dual-base (simultaneous) exponentiation hardware may further increase the overall performance. This is especially important when the overhead caused by communication can be decreased and a large number of attributes are included in the credential.

In future work, the efficiency of communication handover (e.g., from NFC to Bluetooth) could be studied, to decrease the overhead of the communication. More complex credential proofs such as range proofs and credential revocation were not implemented. Future work could be directed to find out the impact of these additional features on the performance of the verification and the requirements of the embedded platform (e.g., connectivity with a revocation server).

References

1. P. Amberg, N. Pinckney, and D.M. Harris. Parallel high-radix montgomery multipliers. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 772 –776, oct. 2008.
2. J.-C. Bajard, L.-S. Didier, and P. Kornerup. An rns montgomery modular multiplication algorithm. In *Computer Arithmetic, 1997. Proceedings., 13th IEEE Symposium on*, pages 234 –239, jul 1997.
3. Patrik Bichsel, Jan Camenisch, Bart De Decker, Jorn Lapon, Vincent Naessens, and Dieter Sommer. Data-minimizing authentication goes mobile. In Bart De Decker and David W. Chadwick, editors, *13th Joint IFIP TC6 and TC11 Conference on Communications and Multimedia Security - CMS 2012 September 3th - September 5th, 2012, Canterbury, UK,*, pages 55–71. Springer, September 2012.
4. Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard java card. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 600–610, New York, NY, USA, 2009. ACM.
5. T. Blum and C. Paar. Montgomery modular exponentiation on reconfigurable hardware. In *Computer Arithmetic, 1999. Proceedings. 14th IEEE Symposium on*, pages 70 –77, 1999.

6. T. Blum and C. Paar. High-radix montgomery modular exponentiation on reconfigurable hardware. *Computers, IEEE Transactions on*, 50(7):759 –764, jul 2001.
7. A. Booth. A Signed Binary Multiplication Technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240, June 1951.
8. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.
9. Ernest Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145. ACM, 2004.
10. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Berlin / Heidelberg, 2001.
11. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Berlin / Heidelberg, 2003.
12. David Chaum. Security without identification: transaction systems to make big brother obsolete. *Commununications of the ACM*, 28(10):1030–1044, October 1985.
13. A. de la Piedra, A. Touhafi, and G. Cornetta. Cryptographic accelerator for 802.15.4 transceivers with key agreement engine based on montgomery arithmetic. In *Communications and Vehicular Technology in the Benelux (SCVT), 2011 18th IEEE Symposium on*, pages 1 –5, nov. 2011.
14. Kurt Dietrich. Anonymous credentials for java enabled platforms: A performance evaluation. In Liqun Chen and Moti Yung, editors, *Trusted Systems*, volume 6163 of *Lecture Notes in Computer Science*, pages 88–103. Springer Berlin Heidelberg, 2010.
15. Tim France-Massay. MULTOS - the high security smart card OS. Technical report, MAOSCO Limited, 2005.
16. Yajuan He and Chip-Hong Chang. A new redundant binary booth encoding for fast 2^n-bit multiplier design. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(6):1192 –1201, june 2009.
17. Specification of the Identity Mixer cryptographic library – version 2.3.2, 2010. IBM Research – Zurich.
18. Y.-P. Lai and C.-C. Chang. An efficient multi-exponentiation scheme based on modified booth's method. *International Journal of Electronics*, 90(3):221 –233, 2003.
19. P. L. Montgomery. Modular multiplication without trail division. *Mathematics of Computation*, 44(170):519–521, 1985.
20. Wojciech Mostowski and Pim Vullers. Efficient u-prove implementation for anonymous credentials on smart cards. In Muttukrishnan Rajarajan, Fred Piper, Haining Wang, and George Kesidis, editors, *Security and Privacy in Communication Networks*, volume 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 243–260. Springer Berlin Heidelberg, 2012.
21. N. Nedjah and L. de Macedo Mourelle. Reconfigurable hardware implementation of montgomery modular multiplication and parallel binary exponentiation. In *Digital System Design, 2002. Proceedings. Euromicro Symposium on*, pages 226 – 233, 2002.

22. N. Nedjah and Ld.M. Mourelle. Three hardware architectures for the binary modular exponentiation: sequential, parallel, and systolic. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 53(3):627 –633, march 2006.
23. J.C. Neto, A.F. Tenca, and W.V. Ruggiero. A parallel k-partition method to perform montgomery multiplication. In *Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on*, pages 251 –254, sept. 2011.
24. S.B. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of a montgomery modular multiplier in a systolic array. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, page 8 pp., april 2003.
25. Geoffrey Ottoy, Jeroen Martens, Nick Saeys, Bart Preneel, Lieven Strycker, Jean-Pierre Goemaere, and Tom Hamelinckx. A Modular Test Platform for Evaluation of Security Protocols in NFC Applications. In Bart Decker, Jorn Lapon, Vincent Naessens, and Andreas Uhl, editors, *Communications and Multimedia Security*, volume 7025 of *Lecture Notes in Computer Science*, pages 171–177. Springer Berlin Heidelberg, 2011.
26. Geoffrey Ottoy, Bart Preneel, Jean-Pierre Goemaere, and Lieven Strycker. Flexible design of a modular simultaneous exponentiation core for embedded platforms. In Philip Brisk, JoséGabriel Figueiredo Coutinho, and PedroC. Diniz, editors, *Reconfigurable Computing: Architectures, Tools and Applications*, volume 7806 of *Lecture Notes in Computer Science*, pages 115–121. Springer Berlin Heidelberg, 2013.
27. Preneel B. Goemaere J.-P. Stevens N. De Strycker L. Ottoy, G. Open-source hardware for embedded security. *EDN*, 2013.
28. B. Phillips. Modular multiplication in the montgomery residue number system. In *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, volume 2, pages 1637 –1640, nov. 2001.
29. N. Pinckney and D.M. Harris. Parallelized radix-4 scalable montgomery multipliers. In Antonio Petraglia, Volnei A. Pedroni, and Gert Cauwenberghs, editors, *Proceedings of the 20th Annual Symposium on Integrated Circuits and Systems Design, SBCCI 2007, Copacabana, Rio de Janeiro, Brazil, September 3-6, 2007*, pages 306–311. ACM, 2007.
30. N. Pinckney and D.M. Harris. Parallelized radix-4 scalable montgomery multipliers. *Journal of Integrated Circuits and Systems*, pages 39 –45, 2008.
31. K. Shigemoto, K. Kawakami, and K. Nakano. Accelerating montgomery modulo multiplication for redundant radix-64k number system on the fpga using dual-port block rams. In *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, volume 1, pages 44 –51, dec. 2008.
32. Christian Paquin Stefan Brands. U-Prove cryptographic specification v1.0, 2010. Microsoft Corporation.
33. Michael Sterckx, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Efficient implementation of anonymous credentials on Java Card smart cards. In *First IEEE International Workshop on Information Forensics and Security, 2009. WIFS 2009*, pages 106 –110, December 2009.
34. G.D. Sutter, J.-P. Deschamps, and J.L. Imana. Modular multiplication and exponentiation architectures for fast rsa cryptosystem based on digit serial computation. *Industrial Electronics, IEEE Transactions on*, 58(7):3101 –3109, july 2011.
35. A.F. Tenca and Ç.K. Koç. A scalable architecture for montgomery multiplication. In *Lecture Notes in Computer Science*, pages 94–108. Springer, 1999.
36. A.F. Tenca and Ç.K. Koç. A scalable architecture for modular multiplication based on montgomery's algorithm. *Computers, IEEE Transactions on*, 52(9):1215 – 1221, sept. 2003.

37. Hendrik Tews and Bart Jacobs. Performance issues of selective disclosure and blinded issuing protocols on java card. In Olivier Markowitch, Angelos Bilas, Jaap-Henk Hoepman, ChrisJ. Mitchell, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks*, volume 5746 of *Lecture Notes in Computer Science*, pages 95–111. Springer Berlin Heidelberg, 2009.
38. Pim Vullers and Gergely Alpár. Efficient selective disclosure on smart cards using Idemix. In Simone Fischer-Hübner and Elisabeth de Leeuw, editor, *3rd IFIP WG 11.6 Working Conference on Policies and Research in Identity Management, IDMAN 2013, London, UK, April 8-9, 2013. Proceedings*, IFIP Advances in Information and Communication Technology. Springer-Verlag, April 2013.
39. Y. Zhou and X. Wang. An improved implementation of montgomery algorithm using efficient pipelining and structured parallelism techniques. In *Signals and Systems Conference (ISSC 2010), IET Irish*, pages 7 –11, june 2010.

A CL-based Credentials

CL *Signature Scheme.* We briefly recall the CL-signature scheme, with a signer S and verifier V, for blocks of L messages as presented in [11] and implemented in the `Identity Mixer` library [17]:

I : $(pk_{Sig}, sk_{Sig}) \leftarrow \mathsf{setup}_{\mathsf{CL}}(l_n)$
Choose a special RSA modulus $n = pq$ of length l_n with $p = 2p'+1, q = 2q'+1$ where p, q, p' and q' are prime. Choose, uniformly at random $R_0, ..., R_{L-1}, S, Z \in_R QR_n$ with public key $pk_{Sig} = (n, R_0, \ldots, R_{L-1}, S, Z)$ and secret key $sk_{Sig} = (p)$.

S : $(\sigma) \leftarrow \mathsf{sign}_{\mathsf{CL}}(m_0, \ldots, m_{L-1}, sk_{Sig})$
Let l_m be a parameter defining the message space as $m_i \in \pm\{0,1\}^{l_m}$ for $0 < i < L$. Choose a random prime e of length $l_e > l_m + 2$ and a random number $v \in_R \pm\{0,1\}^{l_n+l_m+l_r}$, with l_r a security parameter, and compute the signature $\sigma = (A, e, v)$ on $(m_0, ..., m_{L-1})$ such that $A^e \equiv \frac{Z}{R_0^{m_0} ... R_L^{m_{L-1}} S^v}$ mod n. The latter requires knowledge of the order of the subgroup to compute the inverse of e.

V : $(Bool) \leftarrow \mathsf{verify}(\sigma, m_0, \ldots, m_{L-1}, pk_{Sig})$
Parse σ as a tuple (A, e, v) and return true if $Z \equiv A^e R_0^{m_0} \ldots R_{L-1}^{m_{L-1}} S^v \mod n$, $2^{l_e-1} < e < 2^{l_e}$ and $m_i \in \pm\{0,1\}^{l_m}$ for $0 < i \leq L$ holds, else return false.

Note that to fully benefit from the privacy features provided by attribute-based credentials, they should be used in combination with anonymous communication in order to prevent linking or identification through for instance IP address, MAC address, cookies, or browser identification. Nevertheless, even without anonymous communication, attribute-based credentials are superior and more privacy-friendly than traditional authentication technologies.

Signature Proof of Knowledge. Attribute-based credential authentication based on CL signatures (e.g., as in the `Identity Mixer` library), mainly consist of a

signature proof of knowledge of a CL signature $\sigma = (A, e, v)$ on a nonce n_1 as recalled below. Note that the protocols only consider proving knowledge of a valid credential and selective disclosure. For more advanced protocols, such as interval proofs and enumeration, we refer the reader to the `Identity Mixer` specification [17]

$$\begin{aligned} SPK\{&(e, \{m_i : i \in A_h\}, v) : \\ &\frac{Z}{\prod_{i \in A_r} R_i^{m_i}} \equiv \pm A^e S^v \prod_{j \in A_h} R_j^{m_j} \\ &\forall i \in A_h : m_i \in \{0,1\}^{l_m + l_\phi + l_H + 2} \\ &e - 2^{l_e - 1} \in \{0,1\}^{l'_e + l_\phi + l_H + 2} \\ \}&(n_1) \end{aligned} \tag{7}$$

with A_h and A_r the set of hidden, resp. revealed attributes, l_m, l_H, l_e the bit length of the attributes, the challenge and e respectively, l_ϕ a security parameter that governs the statistical zero-knowledgeness and l'_e the size of the interval the e values are taken from. m_0 is called the master secret and is never revealed.

This signature proof of knowledge can be converted into the following two protocols:

Construction of the proof. After receiving the nonce n_1 from the verifier, the prover builds the proof as follows:

1. Randomize CL-Signature $\sigma = (A, e, v)$:
$$\begin{aligned} r_A &\in_R \{0,1\}^{l_n + l_\phi} \\ A' &= AS^{r_A} \mod n \\ v' &= v - er_A, \\ e' &= e - 2^{l_e - 1} \end{aligned}$$
2. Compute 1st round:
$\tilde{e} \in_R \pm\{0,1\}^{l_{e'} + l_\phi + l_H}$, $\tilde{v}' \in_R \pm\{0,1\}^{l_v + l_\phi + l_H}$, $\tilde{m}_i \in_R \{0,1\}^{l_m + l_\phi + l_H}$ $\forall i \in A_h$
$$\tilde{Z} = (A')^{\tilde{e}} \left(\prod_{i \in A_h} R_i^{\tilde{m}_i} \right) (S^{\tilde{v}'}) \mod n$$
3. Compute challenge:
$$c = H(context, A', \tilde{Z}, n_1)$$
4. Compute 2nd round:
$$\begin{aligned} \hat{e} &= \tilde{e} + ce' \\ \hat{v}' &= \tilde{v}' + cv' \\ \hat{m}_i &= \tilde{m}_i + cm_i \ \ \forall i \in A_h \end{aligned}$$

Let the proof $\pi = (c, A', \hat{e}, \hat{v}', \hat{m}_i, m_j \forall i \in A_h \text{ and } \forall j \in A_r)$

Verification of the proof by the verifier. See Sect. 3.