Group-Signature Schemes on Constrained Devices: The Gap Between Theory and Practice

Raphael Spreitzer and Jörn-Marc Schmidt
Graz University of Technology
Inffeldgasse 16a, 8010 Graz, Austria
{raphael.spreitzer, joern-marc.schmidt}@iaik.tugraz.at

ABSTRACT

Group-signature schemes allow members within a predefined group to prove specific properties without revealing more information than necessary. Potential areas of application include electronic IDs (eIDs) and smartcards, *i.e.*, resource-constrained environments. Though literature provides many theoretical proposals for group-signature schemes, practical evaluations regarding the applicability of such mechanisms in resource-constrained environments are missing. In this work, we investigate four different group-signature schemes in terms of mathematical operations, signature length, and the proposed revocation mechanisms. We also use the RELIC toolkit to implement the two most promising of the investigated group-signature schemes—one of which is going to be standardized in ISO/IEC 20008—for the AVR microcontroller. This allows us to give practical insights into the applicability of pairings on the AVR microcontroller in general and the applicability of group-signature schemes in particular on the very same. Contrary to the general recommendation of precomputing and storing pairing evaluations if possible, we observed that the evaluation of pairings might be faster than computations on cached pairings.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

ATmega128, AVR, group-signature schemes, pairing-based cryptography, Type 1 pairings, Type 3 pairings

1. INTRODUCTION

Pairing-based cryptography has lead to the development of new cryptographic protocols, for instance, identity-based encryption [5], identity-based key agreement protocols [20], and group-signature schemes [9]. While identity-based key agreement protocols [14, 19, 23] have already been investigated in resource-constrained environments, investigations regarding the applicability of group-signature schemes in resource-constrained environments are still missing.

Group-signature schemes allow members of a predefined group to sign messages on behalf of the group anonymously. Potential scenarios for such schemes are, for instance, a proof of the age of majority and anonymous entrance control systems. Both scenarios might be implemented on resource-constrained devices, *i.e.*, smartcards or electronic IDs (eIDs). Therefore, the investigation regarding the implementation of such schemes on embedded systems is of major interest for protocol designers as well as developers. However, only minor work has been done in this area of research. For instance, Canard *et al.* [7] modified the proposed signature scheme by Delerablée and Pointcheval [11] in order to perform the signature generation in less than 200 ms on an ATmega128 clocked at 7.37 MHz. However, they adapted the scheme in such a way that the user sacrifices anonymity against a powerful intermediary taking part in the computation of the signature. Considering a scenario where a person wants to prove a specific age to an authority without revealing the exact date of birth, *e.g.*, at a cigarette vending machine, the employment of a powerful intermediary decreases the usability. Furthermore, this approach might only be practical if the used intermediary is something the user has control over.

Our contributions can be summarized as follows. First, we investigate and compare different group-signature schemes based on the number of executed operations, the signature size, and the proposed revocation mechanisms. Second, by employing the RELIC toolkit [2], a state-of-the-art library for pairing-based cryptography, we also implement two of the presented group-signature schemes on the AVR microcontroller. This allows us to compare the performance of two different types of pairings, *i.e.*, Type 1 and Type 3 pairings, in general and to provide insights into the application of pairing-based protocols like group-signature schemes.

2. PRELIMINARIES

Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T$ be cyclic groups of prime order n and their respective generators. Then a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a function, such that $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_n^*$. Furthermore, $e(g_1, g_2) \neq 1$ must hold. According to Galbraith *et al.* [12] four different types of pairings exist:

Type 1. $\mathbb{G}_1 = \mathbb{G}_2$, also known as symmetric pairings.
Type 2. $\mathbb{G}_1 \neq \mathbb{G}_2$, an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ exists, but hashing into $\mathbb{G}_2$ is infeasible.
Type 3. $\mathbb{G}_1 \neq \mathbb{G}_2$, an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ does not exist, but hashing into $\mathbb{G}_2$ is possible.

CS2, January 20 2014, Vienna, Austria.

ACM 978-1-4503-2484-7/14/01 ...$15.00.
http://dx.doi.org/10.1145/2556315.2556321.

Type 4. $\mathbb{G}_1 \neq \mathbb{G}_2$, an efficiently computable homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ exists, and hashing into $\mathbb{G}_2$ is possible. In this case $\mathbb{G}_2$ is a non-cyclic group of order n^2.

Usually $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of points on elliptic curves defined over a finite field $\mathbb{F}_q$ and an extension field $\mathbb{F}_{q^k}$ respectively. $\mathbb{G}_T$ usually represents a subgroup of the multiplicative finite field $\mathbb{F}^*_{q^k}$. The parameter q denotes the size of the underlying field and k denotes the *embedding degree* such that $n|q^k - 1$, with k minimal. Note that the discrete-logarithm problem (DLP) must be hard in all three groups, and thus the group order n, the size of the underlying field q and the embedding degree k must be selected appropriately for a specific security level [12].

3. GROUP-SIGNATURE SCHEMES

In 1991, Chaum and van Heyst [9] proposed the concept of group-signature schemes (GSSs), which provide members of a predefined group the ability to sign messages on behalf of the group. A verifier is able to determine whether a signature was created by an honest member of a specific group, but does not learn the identity of the signer. However, in case of misbehavior a specific party is able to reveal the identity of the signer, *i.e.*, to open the signature. In addition, a specific party should be able to suspend specific users, such that they are not able to generate valid signatures anymore. The involved parties of a group-signature scheme can be summarized as follows:

Signer. A member of a group who is able to sign messages on behalf of the group with the corresponding private key.

Verifier. Given a group's public key, the verifier can determine whether a signature was created by a specific group.

Group manager and group master. An entity in charge of performing the initial setup phase, *i.e.*, to generate the public parameters including the group's public key. Meiklejohn [18] distinguishes between the *group manager* and the *group master*. *Group managers* help joining users to generate their secret keys, but do not learn the actual secret keys. In contrast, *group masters* issue secret keys and, hence, know the secret keys of all group members. Additionally, one of these two parties shall be able to reveal the identity of a signer and to suspend specific users.

Depending on whether or not users can join a group after performing the setup phase, one distinguishes between *dynamic groups* and *static groups*. Nevertheless, static group-signature schemes can be implemented to look like dynamic group-signature schemes by creating signing keys in advance and distribute these keys as new members join the group [18].

Group-signature schemes allow for numerous scenarios that require a proof of a specific property to gain access to specific services and locations. Examples include the proof of a specific age, proof of possession of a valid driving license, anonymous entrance control, and many more. All without the ability to trace specific persons. Another important feature of group-signature schemes is the suspension of specific users. The process of suspending specific users is also referred to as revocation and Boneh and Shacham [6] state three approaches to implement the revocation mechanism.

Perform setup phase again. The group manager generates a new public key and all remaining group members receive a corresponding signing key.

Update private keys. The group members are requested to update their signing keys, but revoked users are not able to compute a valid key anymore.

Verifier-local revocation. *Verifier-local revocation* (VLC) overcomes the rather complicated solution of a recurring setup phase. The verifier is given a list of revoked members and hence the signing capabilities of unrevoked members are not affected. The drawback of this mechanism is that revoked members loose anonymity, which results from the fact that the verifier is able to call the verification algorithm twice, once with the revocation list and once without it. Another disadvantage of this approach is that the complexity of verifying signatures grows with the size of the revocation list. Hence, at some point it might be more feasible to perform the initial setup phase again and to distribute new keys to all remaining group members.

3.1 Investigation of Group-Signature Schemes

In this section, we investigate some of the most recent group-signature schemes. Since we concentrate on scenarios where we are supposed to generate signatures on a resource-constrained device, we investigate the process of generating a signature only. The abbreviations of the following schemes are derived from the authors' names.

3.1.1 BBS

The security of the group-signature scheme by Boneh *et al.* [4] is based on the *strong Diffie-Hellman* (SDH) assumption [3], and the *Decision Linear* Assumption (DLIN). Based on the *Decision Linear* assumption they define Linear Encryption (LE), which is used to hide the user's certificate within a group signature. The hidden certificate can be decrypted by the group manager only and thus can be used to open a signature later on.

The public parameters are a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_n$, $h \in_R \mathbb{G}_1$, $u, v \in \mathbb{G}_1$, $g_2 \in_R \mathbb{G}_2$, $g_1 = \psi(g_2)$, and $w = g_2^{\gamma}$, for a secret γ (only known to the issuer of the signing keys). The user's private key is denoted as (A, x), with $A^{x+\gamma} = g_1$, and $x \in_R \mathbb{Z}_n$. Type 2 pairings are to be used due to the efficiently computable isomorphism ψ. Though one might also use Type 1 pairings[1], in which case $\mathbb{G}_1 = \mathbb{G}_2$.

Signature Generation. The signer chooses α, β, r_α, r_β, r_x, r_{δ_1}, $r_{\delta_2} \in_R \mathbb{Z}_n^*$ and computes the following values based on a specific message M.

$$
\begin{array}{lll}
T_1 = u^{\alpha} & T_2 = v^{\beta} & T_3 = Ah^{\alpha+\beta} \\
\delta_1 = x\alpha & \delta_2 = x\beta & R_1 = u^{r_\alpha} \\
R_2 = v^{r_\beta} & R_4 = T_1^{r_x} u^{-r_{\delta_1}} & R_5 = T_2^{r_x} v^{-r_{\delta_2}} \\
\multicolumn{3}{l}{R_3 = e(T_3, g_2)^{r_x} e(h, w)^{-r_\alpha - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}} \\
\multicolumn{3}{l}{c = H(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)} \\
s_\alpha = r_\alpha + c\alpha & s_\beta = r_\beta + c\beta & s_x = r_x + cx \\
s_{\delta_1} = r_{\delta_1} + c\delta_1 & s_{\delta_2} = r_{\delta_2} + c\delta_2 &
\end{array}
$$

Boneh *et al.* [4] suggest to cache the evaluated pairings $e(h, w)$, $e(h, g_2)$, $e(g_1, g_2)$, and $e(A, g_2)$. Hence, as outlined below, the computation of R_3 can be performed without the need to evaluate a single pairing.

$$
\begin{aligned}
R_3 &= e(T_3, g_2)^{r_x} e(h, w)^{-r_\alpha - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}} \\
&= e(A, g_2)^{r_x} e(h, g_2)^{r_x(\alpha+\beta) - r_{\delta_1} - r_{\delta_2}} e(h, w)^{-r_\alpha - r_\beta}
\end{aligned}
$$

[1]According to Chatterjee *et al.* [8] this protocol cannot be implemented using Type 1 pairings due to the XDH assumption. However, BBS relies on the DLIN assumption and the XDH assumption only allows for even shorter group signatures than presented here [4, p 17].

The signature is $s = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, with T_1, T_2, and T_3 representing the encryption of the user's certificate with respect to the group manager's public key.

Revocation. The revocation requires the computation of a new private key for each user remaining in the group, which necessitates the computation of the isomorphism ψ. The *Revocation Authority* publishes $RL = \{(A_i^*, x_i)\}$, which represents a part of the certificate and the private key of the revoked users, such that $A_i = \psi(A_i^*)$. Given one pair of the revocation list $(A_i^*, x_i) \in RL$ and supposing a user's private key to be (A, x), a new private key $(\hat{A}, x)$ is computed as:

$$\hat{A} = \psi(A_i^*)^{\frac{1}{x - x_i}} \left(A^{\frac{1}{x - x_i}} \right)^{-1}$$

The main drawback of this scheme is that the key issuer is in possession of a list of all private keys. Thus, the key issuer is able to sign messages on behalf of any group member and the property of *framing-resistance* [10] does not hold anymore. However, Boneh *et al.* [4] also proposed a *JOIN* protocol where the user generates the private key and proves correctness of the generated value to the key issuer. In this case the key issuer does not learn the private key anymore.

3.1.2 BS-GS

Boneh and Shacham [6] presented a signature scheme which is based on the *strong Diffie-Hellman* assumption (SDH), and the *Decision Linear* assumption in groups with a bilinear map. The advantage of this signature scheme is that it provides a verifier-local revocation. However, the drawback is the rather complicated process of opening signatures. Therefore, the group manager temporary adds all users, one at a time, to the revocation list and verifies the signature. The first user for whom the signature is invalid is considered to be the signer. Since Chatterjee *et al.* [8] claim this protocol to be insecure due to the revocation check, which might lead to a signature generated by a revoked user being accepted as valid, we state their version of this protocol.

The public parameters are a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, a hash function $H \to \mathbb{Z}_n$, a hash function $H_0 \to \mathbb{G}_2 \times \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $g_1 = \psi(g_2)$, $w = g_2^\gamma$—for a secret γ—and $pgk = (g_1, g_2, w)$. Since hashing into $\mathbb{G}_2 \times \mathbb{G}_1$ and the computation of the isomorphism ψ is necessary, Type 4 pairings are to be used. Each user's private key consists of a tuple (A_i, x_i), with $x_i \in_R \mathbb{Z}_n^*$ and $A_i = g_1^{\frac{1}{\gamma + x_i}}$.

Signature Generation. The signer chooses r, α, r_α, r_x, $r_\delta \in_R \mathbb{Z}_n^*$ and computes the following values based on a specific message M.

$$\begin{array}{lll}
(\hat{u}, v) = H_0(gpk, M, r) & u = \psi(\hat{u}) & \hat{T}_1 = \hat{u}^\alpha \\
T_2 = A_i v^\alpha & \delta = x_i \alpha & R_1 = u^{r_\alpha} \\
\hat{R}_3 = \hat{T}_1^{r_x} \hat{u}^{-r_\delta} & R_2 = e(T_2, g_2)^{r_x} e(v, w^{-r_\alpha} g_2^{-r_\delta}) & \\
c = H(gpk, M, r, \hat{T}_1, T_2, R_1, R_2, \hat{R}_3) & & \\
s_\alpha = r_\alpha + c\alpha & s_x = r_x + cx_i \quad s_\delta = r_\delta + c\delta &
\end{array}$$

The resulting signature is $s = (r, \hat{T}_1, T_2, c, s_\alpha, s_x, s_\delta)$.

Revocation. A revocation list $RL = \{(A_i)\}$ is used to revoke users. Thus, in addition to the verification of the signature a verifier has to check for each element $A \in RL$ whether $e(T_2 A^{-1}, \hat{u}) = e(v, \hat{T}_1)$ holds. In this case, the signature is considered to be invalid because the signer has been revoked.

Signature generation as well as validation requires the computation of the isomorphism ψ. Additionally, the signature generation requires two pairing evaluations and due to the revocation check the signature verification requires $4 + |RL|$ pairing evaluations. Boneh and Shacham also propose a revocation check which is independent of the size of the revocation list RL, but the drawback of this revocation check is that the signer looses part of his anonymity.

3.1.3 DP-XSGS

Delerablée and Pointcheval [11] proposed the so called eXtremely Short Group-Signature scheme (XSGS). Their scheme is dynamic, *i.e.*, allows users to join the group after the initial setup phase. The security is based on the *strong Diffie-Hellman* assumption (SDH) and the *eXternal Diffie-Hellman* assumption (XDH) [4, 11]. If the used groups do not satisfy the XDH assumption then they rely on the DLIN assumption and claim that the signature enlarges a bit.

The public parameters are a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, a hash function $H : \{0,1\}^* \to \mathbb{Z}_n$, $\langle g_2 \rangle \in_R \mathbb{G}_2$, $\langle g, h, k \rangle \in \mathbb{G}_2$, $g_1 = \psi(g_2)$, and $w = g_2^\gamma$, with γ being known only to the group manager. The private key of each user is denoted as (A, x, y), such that $A^{x+\gamma} = g_1 h^y$ with γ being known only to the group manger, and $x, y \in_R \mathbb{Z}_n^*$. Type 2 pairings are to be used due to the computable isomorphism ψ.

Signature Generation. The signer chooses α, β, r_α, r_β, r_x, $r_z \in_R \mathbb{Z}_n^*$ and computes the following values.

$$\begin{array}{llll}
T_1 = k^\alpha & T_2 = Ah^\alpha & T_3 = k^\beta & T_4 = Ag^\beta \\
R_1 = k^{r_\alpha} & R_3 = k^{r_\beta} & R_4 = \frac{h^{r_\alpha}}{g^{r_\beta}} & z = x\alpha + y \\
R_2 = e(T_2, g_2)^{r_x} e(h, w)^{-r_\alpha} e(h, g_2)^{-r_z} & & & \\
c = H(m, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4) & & & \\
s_\alpha = r_\alpha + c\alpha & s_\beta = r_\beta + c\beta & s_x = r_x + cx & s_z = r_z + cz
\end{array}$$

The resulting signature is $s = (T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z)$, with T_1, T_2, T_3, and T_4 representing the encryption of the user's certificate and allow the group manager to determine the identity of the signer.

Revocation. The revocation of a user requires the remaining users to update their private keys, which involves the computation of the isomorphism ψ.

The process of signature generation does not require the evaluation of a pairing because all pairings can be cached. In case one does not cache the pairings, the computation of R_2 might be collapsed as $R_2 = e(T_2, g_2)^{r_x} e(h, w^{-r_\alpha} g_2^{-r_z})$.

3.1.4 HLCCN

The scheme by Hwang *et al.* [15] is based on the *external Diffie-Hellman* (XDH) assumption, and the *modified q-Strong Diffie-Hellman* (SDH^+) assumption. A *linking key* allows to determine whether two signatures stem from the same signer, but the identity of the signer remains secret.

The public parameters are a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, a hash function $H : \{0,1\}^* \to \mathbb{Z}_n$, $g, u, g_1, g_2 \in_R \mathbb{G}_1$, $h_1 \in_R \mathbb{G}_2$, $w = u^\eta$, $d = u^\xi$, and $h_\theta = h_1^\theta$, with $\theta \in_R \mathbb{Z}_n^*$ being the master's issuing key and $\eta, \xi \in \mathbb{Z}_n^*$ being the master's opening key. The user's signing key is (A, x, y, z), with $A \in \mathbb{G}_1$, and $x, y, z \in \mathbb{Z}_n^*$. Since a computable isomorphism ψ is not required for this scheme, Type 3 pairings can be used.

Table 1: Comparison of the presented group-signature schemes

		BBS [4]	**BS-GS** [6, 8]	**DP-XSGS** [11]	**HLCCN** [15]
Assumptions		SDH, DLIN	SDH, DLIN	SDH, XDH	SDH^+
Revocation		Key update	Verifier local	Key update	Key update
Isomorphism required		Yes	Yes	Yes	No
Hashing into $\mathbb{G}_2$ required		No	Yes	No	No
Pairing		Type 1, 2	Type 4	Type 2	Type 3
Public parameters	$\mathbb{G}_1$	4	1	4	6
	$\mathbb{G}_2$	2 [a]	2	2	2
	$\mathbb{G}_T$	3	0	3	4
Private key	$\mathbb{Z}_n^*$	1	1	2	3
	$\mathbb{G}_1$	1	1	1	1
Sign	R in $\mathbb{Z}_n^*$	7	5	6	5
	A in $\mathbb{Z}_n^*$	9	3	6	5
	M in $\mathbb{Z}_n^*$	8	4	6	6
	M in $\mathbb{G}_1$	3	1	3	3
	E in $\mathbb{G}_1$	9	2	8	7
	M in $\mathbb{G}_2$	0	2	0	0
	E in $\mathbb{G}_2$	0	5	0	0
	M in $\mathbb{G}_T$	2	1	2	4
	E in $\mathbb{G}_T$	3	1	3	5
	Pairing	0	2	0	0
	Isomorphism ψ	0	1	0	0
	Hashing into $\mathbb{G}_2$	0	1	0	0
Verify	R in $\mathbb{Z}_n^*$	0	0	0	0
	A in $\mathbb{Z}_n^*$	2	1	0	0
	M in $\mathbb{Z}_n^*$	0	0	0	0
	M in $\mathbb{G}_1$	4	1	5	3
	E in $\mathbb{G}_1$	8	3	7	5
	M in $\mathbb{G}_2$	1	3	1	1
	E in $\mathbb{G}_2$	2	6	2	2
	M in $\mathbb{G}_T$	3	$2 + \lvert\text{RL}\rvert$ [b]	3	4
	E in $\mathbb{G}_T$	3	1	3	4
	Pairing	1	$4 + \lvert\text{RL}\rvert$	1	1
	Isomorphism ψ	0	2	0	0
	Hashing into $\mathbb{G}_2$	0	1	0	0
Signature size	$\mathbb{Z}_n^*$	6	5	5	5
	$\mathbb{G}_1$	3	1	4	3
	$\mathbb{G}_2$	0	1	0	0

[a] In case $\mathbb{G}_1 = \mathbb{G}_2$ the number of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ as well as the number of operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ is accumulated. Though, in case $g_1 = \psi(g_2)$ this might be counted as one element.
[b] $\lvert\text{RL}\rvert$ denotes the size of the revocation list.

Signature Generation. The signer chooses α, r_α, r_x, r_γ, $r_y \in_R \mathbb{Z}_n^*$ and computes the following values.

$$D_1 = u^\alpha \qquad D_2 = Aw^\alpha \qquad D_3 = g^y d^\alpha \qquad \gamma = x\alpha - z$$
$$R_1 = u^{r_\alpha} \qquad R_3 = g^{r_y} d^{r_\alpha}$$
$$R_2 = e(D_2, h_1)^{r_x} e(w, h_\theta)^{-r_\alpha} e(w, h_1)^{-r_\gamma} e(g_2, h_1)^{r_y}$$
$$c = H(M, D_1, D_2, D_3, R_1, R_2, R_3)$$
$$s_\alpha = r_\alpha + c\alpha \quad s_x = r_x + cx \quad s_\gamma = r_\gamma + c\gamma \quad s_y = r_y + cy$$

The resulting signature is $s = (D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$.

Revocation. The revocation requires each user to update his private key. The process of signature generation does not require the evaluation of a pairing because all pairings can be cached. In case one does not cache the pairings, the computation of R_2 might be collapsed as $R_2 = e(D_2^{r_x} w^{-r_\gamma} g_2^{r_y}, h_1) e(w^{-r_\alpha}, h_\theta)$.

3.1.5 Comparison of the Group-Signature Schemes

Table 1 provides a comprehensive comparison of the above presented group-signature schemes, including the number of group operations for the signature generation and verification. The executed operations are abbreviated with R for selecting a random element, A for addition, M for multiplication, E for exponentiation, and *pairing* for the evaluation of a pairing. Note that if $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of points on the elliptic curve $E(\mathbb{F}_q)$ and $E(\mathbb{F}_{q^k})$, the multiplication (M) denotes the addition of two points and the exponentiation (E) denotes the multiplication of a point with a scalar. Further enhancements regarding the implementation of the protocol might be achieved by considering multi-exponentiation techniques, *i.e.*, the simultaneous multiplication of two points with a scalar followed by the addition of the resulting two points. The size for public parameters, private keys, and resulting signatures are expressed in terms of elements of specific groups since the precise size depends on the chosen curve.

4. PRACTICAL IMPLEMENTATION

We employed the RELIC[2] toolkit [2] for the evaluation of group-signature schemes on the AVR microcontroller. A slightly modified implementation of SimulAVR [1] was used to determine cycle-accurate execution times on a modified ATmega128 with 16 KB of RAM.

RELIC provides low-level implementations for binary-field arithmetic as well as for prime-field arithmetic to be used on the AVR. Based on these low-level implementations different curves can be used for Type 1 and Type 3 pairings, respectively. For Type 1 pairings RELIC implements the η_T (eta-t) pairing over supersingular curves with an em-

[2] We used `avr-gcc-4.5.3` to compile of RELIC (rev. 1460).

Table 2: Performance of arithmetic operations

	Type 1 pairing over $\mathbb{F}_{2^m}$		Type 3 pairing over $\mathbb{F}_p$
	$m = 271$	$m = 353$	158-bit prime
A in $\mathbb{G}_1$	284 000	439 000	267 000
M in $\mathbb{G}_1$	$4.8 \cdot 10^6$	$8.5 \cdot 10^6$	$6.5 \cdot 10^6$
A in $\mathbb{G}_2$	284 000	439 000	388 000
M in $\mathbb{G}_2$	$4.8 \cdot 10^6$	$8.5 \cdot 10^6$	$31.3 \cdot 10^6$
M in $\mathbb{G}_T$	111 000	170 000	438 000
E in $\mathbb{G}_T$	$7.9 \cdot 10^6$	$15.2 \cdot 10^6$	$83.2 \cdot 10^6$
Pairing	$14.0 \cdot 10^6$	$27.1 \cdot 10^6$	$62.7 \cdot 10^6$

bedding degree $k = 4$. For the AVR the supported curves are $E(\mathbb{F}_{2^{271}})$ with the irreducible polynomial $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$ and $E(\mathbb{F}_{2^{353}})$ with the irreducible polynomial $f(x) = x^{353} + x^{69} + 1$. For Type 3 pairings RELIC implements the optimal ate pairing over non-supersingular BN-curves with an embedding degree $k = 12$, defined over a 158-bit prime field $\mathbb{F}_p$. In case of BN-curves the prime modulus p of the underlying field $\mathbb{F}_p$ can be parameterized as $p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$. The 158-bit prime curve—denoted as BN-158 curve—is parameterized by $x =$ `0x4000000031` in base 16.

4.1 Evaluation of Type 1 and Type 3 Pairings

As already pointed out by Galbraith *et al.* [12], Type 3 pairings offer good performance and flexibility for high-security parameters. Furthermore, Scott [21] claims that Type 1 (symmetric) and Type 3 (asymmetric) pairings are the most practical ones, thus we stick to these types. However, in 2013 the discrete-logarithm problem (DLP) over binary finite fields (and finite fields of small characteristic in general) has been attacked seriously [13, 17]. A comprehensive overview of these technical advances can be found in [22]. Smart [22] also claims this being the "death knell for pairing based cryptography on Type-1 curves". Hence, these attacks should be kept in mind when considering the implementation of Type 1 pairings. Before the announcement of these attacks, Type 1 pairings implemented on supersingular curves over $\mathbb{F}_{2^{271}}$ and $\mathbb{F}_{2^{353}}$ were considered to provide a security level of 64 and 80 bits, respectively. Considering the recent attacks these security levels might be revised accordingly, and hence we provide the performance evaluation for Type 1 pairings only for the sake of completeness. Nevertheless, Type 3 pairings implemented over the BN-158 curve still provide 78 bits of security.

Table 2 outlines the performance evaluation of Type 1 pairings employing two different binary fields, *i.e.*, $\mathbb{F}_{2^{271}}$ and $\mathbb{F}_{2^{353}}$, and the performance of Type 3 pairings employing a 158-bit prime field $\mathbb{F}_p$. Note that for Type 1 pairings the performance of operations in $\mathbb{G}_1$ is equal to the performance of operations in $\mathbb{G}_2$ since both groups are the same, *i.e.*, symmetric. Recall that now we are talking about $\mathbb{G}_1$ and $\mathbb{G}_2$ being subgroups of points on elliptic curves defined over the finite field $\mathbb{F}_q$ and an extension field $\mathbb{F}_{q^k}$, respectively. Thus, the addition in $\mathbb{G}_1$ and $\mathbb{G}_2$ corresponds to the multiplication (M) of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ within the group-signature schemes outlined in Section 3. Furthermore, the multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$ corresponds to the exponentiation (E) of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ within the group-signature schemes.

Figure 1 illustrates the performance evaluation for the most time-consuming operations. According to this figure one would prefer Type 1 pairings due to the better performance. Szczechowiak *et al.* [23] also suggest the usage of

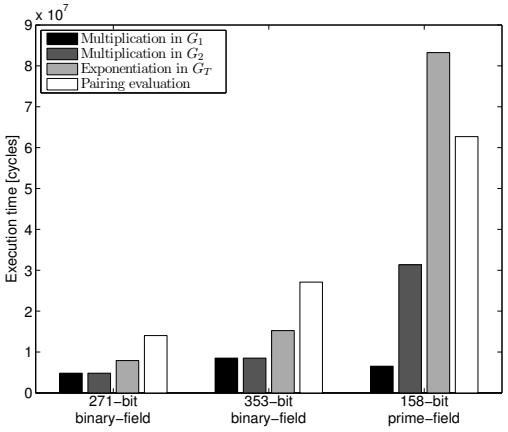

Figure 1: Performance evaluation of Type 1 pairings over $\mathbb{F}_{2^{271}}$ and $\mathbb{F}_{2^{353}}$, and Type 3 pairings over a 158-bit prime field $\mathbb{F}_p$.

Type 1 pairings over $\mathbb{F}_{2^{271}}$ due to the faster computations. However, as already pointed out above, the latest advances in attacks on the DLP in characteristic 2 finite fields emphasize potential security issues in Type 1 pairings in general and, thus, Type 3 pairings seem to be the desirable choice for pairing implementations.

4.2 Possible Performance Optimization

Considering the time-consuming operations in Table 2 let us recall that the computation of a pairing, for instance, $e(u, v)^a$, with u, v, and a known in advance, might be done in multiple different ways:

- Perform an exponentiation in $\mathbb{G}_1$ and evaluate the pairing afterwards: $e(u^a, v)$.
- Perform an exponentiation in $\mathbb{G}_2$ and evaluate the pairing afterwards: $e(u, v^a)$.
- Cache the evaluated pairing and perform an exponentiation in $\mathbb{G}_T$: $e(u, v)^a$.

Given the two alternatives of performing an exponentiation either in $\mathbb{G}_1$ or $\mathbb{G}_2$ one would prefer the former, since usually the exponentiation in the smaller group of $\mathbb{G}_1$ is faster than the exponentiation in $\mathbb{G}_2$. Though protocol designers [4, 11, 15] suggest to cache evaluated pairings due to performance reasons, the presented results indicate that the evaluation of pairings yields faster timings, at least in case of asymmetric pairings. Thus, we conclude that the process of signature generation might be enhanced by performing an exponentiation in $\mathbb{G}_1$, followed by the evaluation of the pairing. Additionally, the evaluation of pairings reduces the storage required for the public parameters, since the evaluated pairings, *i.e.*, elements in $\mathbb{G}_T$, are not cached anymore.

4.3 Comparison of BBS and HLCCN

We implemented the BBS [4] and the HLCCN[3] [15] group-signature schemes since these can be implemented using Type 1 and Type 3 pairings, respectively. This allows us to compare the performance of group-signature schemes implemented over symmetric pairings and asymmetric pairings.

[3]The HLCCN scheme is going to become an ISO standard, specified in *ISO/IEC 20008 Information technology - Security techniques - Anonymous digital signatures* [16].

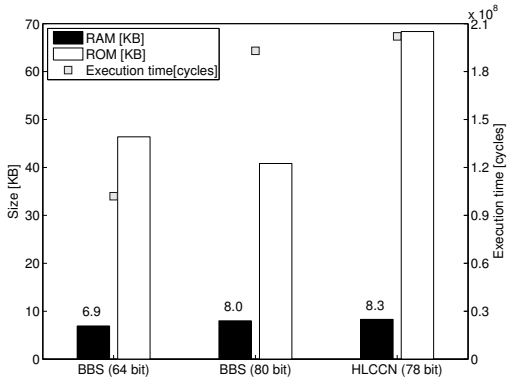

Figure 2: Overall RAM and ROM usage as well as execution time of the two investigated schemes.

Due to the observed timings in Subsection 4.1 we implemented the BBS scheme employing cached pairings and we evaluate the pairings in case of the HLCCN scheme. Figure 2 illustrates the overall execution time in cycles as well as the memory footprint for each of the two investigated group-signature schemes. Considering a clock frequency of 7.4 MHz this would result in about 27 seconds for the signature generation in case of the HLCCN signature scheme. However, considering this scheme being implemented on an ATxmega with a clock frequency of 32 MHz this yields execution times of about 6 seconds and might be even faster due to the ATxmega being a faster microcontroller than the ATmega128. Nevertheless, performance optimizations are necessary in order to achieve acceptable execution times of such complex protocols on the AVR. Besides the introduction of powerful intermediaries as suggested by Canard *et al.* [7], employing instruction-set extensions for the finite-field arithmetic might also lead to acceptable execution times.

5. CONCLUSION

In this work, we compared four different group-signature schemes and provided practical insights into the implementation of group-signature schemes on the AVR microcontroller. Using the RELIC toolkit, we observed that in case of Type 3 pairings the evaluation of a pairing is faster than performing an exponentiation on a cached pairing. Thus, contrary to the suggestion of protocol designers, protocols might gain a performance speedup if pairings are evaluated instead of being cached. Furthermore, our practical observations showed that the ATmega128—even clocked at 16 MHz—might not be able to handle such complex cryptographic protocols under reasonable timing constraints. Even on the ATxmega256, which might be clocked at 32 MHz the HLCCN scheme would require about 6 seconds for the computation of a signature, which is still too long to be considered practicable. Hence, in order to speed up the evaluation of pairings and to facilitate more complex protocols in such resource-constrained environments one might consider to implement finite-field arithmetic in hardware.

Acknowledgements

This work has been supported by the Austrian Government through the research program FIT-IT under the grant number 835917 (NewP@ss). The authors would also like to thank D. F. Aranha for the support regarding RELIC.

6. REFERENCES

[1] Simulavr: an AVR simulator. `http://savannah.nongnu.org/projects/simulavr`.

[2] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.

[3] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT 2004*, volume 3027, pages 56–73. Springer Berlin Heidelberg, 2004.

[4] D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer Berlin Heidelberg, 2004.

[5] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer Berlin Heidelberg, 2001.

[6] D. Boneh and H. Shacham. Group Signatures with Verifier-Local Revocation. In *CCS '04*, pages 168–177, New York, NY, USA, 2004. ACM.

[7] S. Canard, I. Coisel, G. Meulenaer, and O. Pereira. Group Signatures are Suitable for Constrained Devices. In *ICISC 2010*, volume 6829 of *LNCS*, pages 133–150. Springer Berlin Heidelberg, 2011.

[8] S. Chatterjee, D. Hankerson, and A. Menezes. On the Efficiency and Security of Pairing-Based Protocols in the Type 1 and Type 4 Settings. In *Arithmetic of Finite Fields*, volume 6087 of *LNCS*, pages 114–134. Springer Berlin Heidelberg, 2010.

[9] D. Chaum and E. van Heyst. Group Signatures. In *EUROCRYPT '91*, volume 547 of *LNCS*, pages 257–265. Springer Berlin Heidelberg, 1991.

[10] L. Chen and T. Pedersen. New group signature schemes. In *EUROCRYPT '94*, volume 950 of *LNCS*, pages 171–181. Springer Berlin Heidelberg, 1995.

[11] C. Delerablée and D. Pointcheval. Dynamic Fully Anonymous Short Group Signatures. In *VIETCRYPT*, volume 4341 of *LNCS*, pages 193–210, 2006.

[12] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[13] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. On the Function Field Sieve and the Impact of Higher Splitting Probabilities: Application to Discrete Logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. Cryptology ePrint Archive, Report 2013/074, 2013. `http://eprint.iacr.org/`.

[14] C. Gouvêa, L. Oliveira, and J. López. Efficient Software Implementation of Public-Key Cryptography on Sensor Networks Using the MSP430X Microcontroller. *Journal of Cryptographic Engineering*, 2(1):19–29, 2012.

[15] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Short Group Signatures with Controllable Linkability. In *LIGHTSEC '11*, LIGHTSEC '11, pages 44–52, Washington, DC, USA, 2011. IEEE Computer Society.

[16] Internationl Organization for Standardization (ISO). ISO/IEC 20008-2: Information technology - Security techniques - Anonymous digital signatures - Part 2: Mechanisms using a group public key, November 2012.

[17] A. Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. Cryptology ePrint Archive, Report 2013/095, 2013. `http://eprint.iacr.org/`.

[18] S. Meiklejohn. An Exploration of Group and Ring Signatures. Available online at `http://cseweb.ucsd.edu/~smeiklejohn/`, February 2011.

[19] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34(3):485–493, 2011.

[20] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. 2000.

[21] M. Scott. On the Efficient Implementation of Pairing-Based Protocols. In *Cryptography and Coding*, volume 7089 of *LNCS*, pages 296–308. Springer Berlin Heidelberg, 2011.

[22] N. Smart. Discrete Logarithms. `http://bristolcrypto.blogspot.co.uk/2013/02/discrete-logarithms.html`.

[23] P. Szczechowiak, A. Kargl, M. Scott, and M. Collier. On the Application of Pairing Based Cryptography to Wireless Sensor Networks. In *WISEC*, pages 1–12. ACM, 2009.