Smart-card Deployment of an Electronic Voting Protocol

Hervé Chabanne[1,2], Emmanuelle Dottax[1] and Franck Rondepierre[1]
[1]*Idemia, France*
[2]*Télécom ParisTech, France*

Keywords: e-Voting, Smart-cards, Biometrics, Belenios.

Abstract: We present a solution to securely deploy e-voting protocols on the field, thanks to smart-cards. Voters credentials are securely stored on the cards, and the access is restricted by Match-on-Card biometrics. Interestingly, the biometrics verification is made against a list of eligible voters, which allows to restrict the number of cards to one per voting booth. In contrast with previous e-voting solutions requiring a secure element per voter, this constitutes an affordable solution. As an example, we describe how the voting scheme Belenios can be implemented. We show that the resulting scheme gains receipt-freeness, and we give an implementation report that shows the practicability of our solution.

1 INTRODUCTION

Different electronic voting (e-voting) systems have been proposed together with their relative security properties. Ballot privacy (secrecy of the vote) and verifiability (capacity of checking that the election produces the right result) are required. We here describe an implementation of such an e-voting system based on smart-cards – named hereafter *voting cards*.

Our proposal takes advantage of the trust we can put in voting cards. Indeed, the security of smart-cards is today assessed by well-established evaluation processes (as e.g. (Common Criteria, 2018)), and they have already been identified as security enablers in implementing e-voting schemes. However, all proposals made until now assume one-card per voter, and hence are facing cost and deployment difficulties for large scale elections.

In our solution, voting cards are used as cryptographic resources, but only one per voting booth is required. And for a given voting station, each of its voting booth will receive the same voting card. Assuming the polling station takes N voters, any of this N voters will be able to use these voting cards, but no one else. To enforce this policy, the voting cards will perform a biometrics *Match-on-Card (MoC)* against the list of eligible voters. The idea of MoC is that reference biometrics are securely stored on the card, and that the comparison with the live biometrics (the match) is performed inside the card. If the comparison is made against one reference (i.e. authentication), it is called 1:1-MoC, and if N references are compared (i.e. identification), it is called 1:N-MoC.

There are today smart-cards with 1:N-MoC (NIST, 2011) deployed at national level but the idea of performing 1:N-MoC is, to the best of our knowledge, new for voting systems (see Sect. 2.2). It allows our solution to enjoy the security benefits of smart-cards at a limited cost.

Despite its importance, as much we are aware of, we are the first to provide performances (see Sect. 5) for the Belenios e-voting system. We provide figures for both finite field and elliptic curve based computations.

Another interesting feature of our solution is its convenience. Indeed, all the material can be prepared before the election and the process in the voting booth is very light, making smoothness a key feature of the proposal (see Sect. 3).

Finally, we want to stress out the fact that, in our approach, we do not modify the underlying cryptographic e-voting protocol. Rather, we exhibit an economically viable deployment of it (for a broader perspective on the additional economical cost of security systems in modern applications, see (Sklavos and Souras, 2006)). As a comparison, in (Chaidos et al., 2016a), the benefits of secure hardware is pointed out but in the presentation of this very work (Chaidos et al., 2016b), it is indicated that giving a card to each voter would be considered as too expensive. By not changing the cryptographic specifications of the underlying protocol, we aim at keeping most of its security properties. For the specific case of Belenios, quoting (Chaidos et al., 2016a) again, we note that we get the additional receipt-freeness security property with our implementation (See Sect. 4).

Chabanne, H., Dottax, E. and Rondepierre, F.
Smart-card Deployment of an Electronic Voting Protocol.
DOI: 10.5220/0007443805030510
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 503-510
ISBN: 978-989-758-359-9

Related Works. A first version of Belenios appears in (Cortier et al., 2014). The current release of the software can be found online (Belenios, 2018b), as well as a voting platform (Belenios, 2018a). This protocol has been used in more than 200 local elections. The initial protocol is extended in (Chaidos et al., 2016a) to enable receipt-freeness. This is done at the price of relying on more complex cryptographic primitives. A more detailed description is given in Sect. 2.1.

There have been several e-voting system proposals coming with tamper-resistant devices. Proposals to use smart-cards in the intent of achieving receipt-freeness have been made in (Magkos et al., 2001; Lee and Kim, 2002). Also, (Budurushi et al., 2012) provides a survey of the use of national electronic ID-card for remote e-voting in Estonia, Austria and Germany, and for polling station e-voting in Finland. More information on Estonia solution can be found online (Estonia, 2018). Smart-card-based implementations of the Civitas e-voting protocol (Clarkson et al., 2008) are proposed in (Neumann and Volkamer, 2012; Neumann et al., 2013).

In all these solutions, it is assumed that each voter is provided with a personal, tamper-resistant voting device.

2 BACKGROUND

2.1 Belenios

Belenios is an extension of Helios (Adida, 2008; Adida et al., 2009), with an explicit registration authority: eligibility is guaranteed by signing the ballots. It also uses different zero-knowledge proofs. It is close to Helios-C (Cortier et al., 2014) too, but without the threshold support for the decryption of the ballots. The description given here is compliant with the one found in the recent paper (Cortier et al., 2018). More details can be found in the specification (Glondu, 2018).

The main parties involved in the election are:

- A voting server S, representing the bulletin board manager and operated by the server administrator A. It is responsible for processing ballots in the bulletin box BB.
- A credential authority C, who is responsible only for the generation of credentials and their distribution to the voters.
- A set of trustees $T_1, \cdots, T_m$. They are the public authorities in charge of tallying and publishing the result of the election. They also produce the parameters of the election.
- The eligible voters $V_1, \cdots, V_n$.

The following algorithms are used:

Setup(1^λ)**:** The administrator A and the trustees generate the election encryption public key pk, and trustees secret keys $(sk_i)_{(i=1\ldots m)}$ via a distributed cryptosystem parametrized with security parameter λ. Public key pk is an implicit input for all remaining algorithms.

Register(id)**:** Creates a signing key pair (upk_{id}, usk_{id}) for the voter id. This step is run by C. The list L of the public keys of all eligible voters is published, and it is considered an implicit input for all remaining algorithms.

Vote$(id, v, upk_{id}, usk_{id})$**:** Constructs a ballot for the voter id and vote v. The ballot is of the form (id, upk_{id}, c, s), where:

- c is the vote encrypted under the election key pk, together with proofs that the vote is correctly formed,
- s is the voter signature on c, computed with the signing key usk_{id}.

This step is performed by voters willing to express their vote.

Valid(BB, b, pk)**:** Checks the validity of the ballot with respect to the ballot box BB: verifies the signature and the proof of $b = (id, upk_{id}, c, s)$, and also checks there is no same identity id with different credential upk_{id} and no different identity with same credential. This is performed by the voting server S whenever a new vote is received.

Box(BB, b)**:** The voting server S returns the ballot box $\mathsf{BB} \cup b$ if Valid(BB, b, pk) holds, BB otherwise. The ballot box is kept internally.

Publish(FBB)**:** Returns the public views of BB:

- provides the talliers with FBB, a version of BB with voters identities discarded (ballots are now of the form (upk_{id}, c, s)),
- extend each ballot with a hash of it, and publish the result $\mathsf{PBB} = \{(b, hash(b)) | b \in \mathsf{FBB}\}$.

This step is handled by the administrator A.

Tally$(\mathsf{FBB}, (sk_i)_{(i=1\ldots m)})$**:** Returns the result of the election r, as well as a proof π that the tally was computed correctly. To do so, the ballots signature and proofs are verified and invalid ballots are discarded. Then, ciphertexts are added and the sum is decrypted (thanks to homomorphic feature of the encryption algorithm) to get the election result. This is performed conjointly by Administrator A and trustees.

Verify(pk, PBB, r, π)**:** This algorithm allows one to check that the tally corresponds to the public bulletin board FBB via verification of the proof π.

VerifyVote(b, PBB)**:** Checks that the ballot b belongs to the public bulletin board. It is intended to the voters, for checking that their ballot is indeed taken into account.

The security of voting schemes is a vast subject that has attracted a lot of attention. Informally stated, the key properties to achieve are ballot privacy (no one can gain information on the choices of the voters), completeness (all valid votes are counted), eligibility (no one who is not allowed to vote can vote), soundness (any invalid vote is not counted).

For an extensive-review of theses properties and the schemes that offer them, we refer to, e.g., (Benaloh et al., 2017). Machine-checked proofs in EasyCrypt for privacy and verifiability in Belenios are provided in (Cortier et al., 2018).

As Helios, Belenios does not achieve receipt-freeness (a voter has no means to prove how she voted). Indeed, the voter can record and reveal the randomness used to encrypt her ballot. It is then possible to re-encrypt the claimed vote and to check the presence of the corresponding ballot in the public bulletin board (see Sect. 5 for details regarding the encryption process). A receipt-free version of Belenios, called BeleniosRF, has been proposed in (Chaidos et al., 2016a).

2.2 1:N-MoC

Biometric traits can be represented as vectors, known as *templates*, and can be further compared. To measure the performance of a biometric system, one usually considers its False Acceptance Rate (FAR) – the proportion of impostors being wrongly accepted – and False Rejection Rate (FRR) – the proportion of genuine users being rejected erroneously. We here give figures for different modalities: 1:1-MoC for fingeprint and 1:N-MoC for iris and face.

The deep convolutional network Facenet (Schroff et al., 2015) outputs 128-byte vectors embedded in an Euclidean space for coding face images. Two templates match whenever their euclidean distance (l_2-norm) is below a given threshold. An accuracy recognition performance as high as $99.63\% = 1-\text{FRR}$ can then be reached (on the dataset named Labeled Faces in the Wild (Huang et al., 2008)).

As far as 1:1-MoC is concerned, the Minex program (NIST, 2011) by NIST is dedicated to the evaluation of fingerprint minutia matchers running on smart-cards. It reports, for instance, a solution with fingerprints coded on 512 bytes, and with (FRR,FAR) equal to $(0.0047, 0.01)$ or $(0.0086, 0.001)$.

In what follows, we consider that voting cards contain a set of such reference templates for identifying eligible voters. Instead of performing brutal search by performing N matches against the freshly captured biometric data, a better strategy is first to check whether this latter is close enough to one vector within the set. This problem is known in the literature as Nearest Neighbour Search. As an example of what has already been done in this field (Hao et al., 2008), we want to describe a method based on Locality Sensitive Hashing (LSH). The definition of LSH is recalled:

Definition 1. *Let B be a metric space, U a set with smaller dimensionality, $r_1, r_2 \in \mathbb{R}$ with $r_1 < r_2$ and $p_1, p_2 \in [0,1]$ with $p_1 > p_2$.*

A family $H = \{H_1, \ldots, H_k\}, H_j : B \to U$ is (r_1, r_2, p_2, p_2)-LSH if for all $h \in H, x, x' \in B$, $Pr[h(x) = h(x')] > p_1$ if $d_B(x,x') < r_1$ and $Pr[h(x) = h(x')] < p_2$ if $d_B(x,x') > r_2$.

That definition, together with the fact that two biometric traits measurements coming from the same person (resp. from different persons) are close (resp. are far) in the underlying distance, gives to (Hao et al., 2008) the basis of its identification algorithm, the *single collision principle*. The principle can be stated as: two similar biometric records will have high chance to have "colliding" H_j. The identification proceeds by first looking at collisions through LSH functions between the fresh captured biometric data and all the references stored, and then, for the most likely candidates, by checking if a match actually happens. Identification in (Hao et al., 2008) is made against a large iris database. Iris templates are 256-byte long vectors embedded in the Hamming space, and the LSH functions are simply projections over random coordinates.

The same "single collision principle" is used in (Bringer et al., 2009), where a facial MoC implementation is considered. The smart-card here serves to enforce the privacy inside a biometric terminal.

In the case of templates encoded as vectors in the Euclidean space, LSH can be thought as a random projection where each of the H_j is associated with a random line in a 2-dimensional space.

3 BELENIOSHW

In this section, we describe our solution to securely deploy the Belenios system with the following tweaks:

- the replacement of the login/password used for voter authentication in Belenios by a 1:N-MoC,
- the provisioning of voters credentials in a smart-card.

Election Preparation. The election is prepared by performing the following steps:

- Enrolment: biometric templates of eligible voters are gathered.
- Smart-cards provisioning: for each polling station, a set of identical cards is prepared (one per voting booth and some spare). Each card contains the templates of all the voters of the polling station, and for each voter id, his $template_{id}$ is associated with his secret credential usk_{id} – the same as the one used in standard Belenios. Each card also contains the public key pk of the election, used to encrypt the ballots.
- Material deployment: Each polling station is equipped with some terminals, and the set of cards.

Remark 1. *Each voting card can come equipped with its own storage keys. These (symmetric) keys will serve to securely store (confidentiality, integrity) data outside the card. More specifically, all the data needed by a voter to vote, i.e. his cryptographic keys along with some pre-computed data which do not depend on his vote (see Remark 4) can be placed outside the voting cards. Whenever a voter has identified himself, the card will retrieve all the data this very voter needs to vote. This solution can be envisaged as a trade-off to increase the storage capacity of the voting card, at the cost of little more computations.*

Election. We propose to supply each voting booth with a terminal with the adequate interfaces for the voter, the required biometric sensor(s), and a voting card. Once the voter has entered a voting booth in his polling station, his journey is the following:

- The voter presents his biometric data, and the 1:N-MoC is performed;
- If the MoC is successful, the voter is prompted by the terminal to make his choice according to the election;
- The choice is sent to the card, the card builds the ballot by computing the encryption, the proofs and the signature;
- The ballot is sent to the terminal.

Remark 2. *As computations are performed by the smart-card connected to a terminal, the voter has no mean per-se to get his ballot value, and hence no mean to check its presence in the public bulletin board. We propose to rely on a secure interface to display the hash value of his ballot to the voter. For more convenience, an extra printer can also be added in the voting booth to print a paper with the hash value of the ballot. The voter can easily check that the printed value is the same than the one printed. Moreover, doing so, the voter can use it later to check his ballot has been cast.*

The rest of the process – the validation of the ballots by the voting server, the publication of the ballots and the tally by the trustees – is performed exactly as in Belenios.

4 SECURITY PROPERTIES

The variant of Belenios we propose only makes few changes to the original scheme. We here give a detailed view of the modifications, using the notation introduced in Sec. 2.1, and we discuss the impact of these changes on the voting scheme.

Set-up: This step is left unchanged.

Register: This step is slightly modified. The credential authority still generates the signing keys for the voters, but instead of sending them to the voters, these secrets are securely stored in smart-cards, each key being associated to the biometric template of the corresponding voter. The cards are distributed in the polling stations.

Once the cards are personalized this way, the credential authority publishes the list of public keys and forgets the secrets, like in standard Belenios.

Note that this step could be managed such that only the cards know the secrets. It is indeed possible to generate the signing key pairs inside one card, and to securely communicate the secrets to the other cards of the same polling station. Note also that in the new scheme, the voters have no access to the value of their secrets.

Vote: This step is exactly the same as in Belenios, except that the computations are made by a smart-card instead of a software client on voter's computer. There is however a difference. Belenios authenticates voters via a login/password procedure, and voters enter their secret key. Our solution enforces voter authentication via biometrics and hence have the nice feature of preventing a dishonest voter to sell his vote by giving away his password. In addition, as computations are made in smart-cards, nobody – including the voter – has access to intermediate results. In particular, the voter has no mean to learn the details of the encryption computation.

Valid, Box, Publish, Tally, Verify: These steps are left unchanged.

VerifyVote: The voter uses the printed hash value of the ballot to check his ballot has been cast (see Remark 2).

As can be seen, we make no additional, specific assumption on the parties involved in the process:

- On the trustees side, there is strictly no change.
- No change on the Voting Server side, either.
- The Credential authority behaves the same and has access to the same information. Only, sending the keys to the voters is replaced by associating a voter's key to his biometrics in a smart-card. There is no new possibility for this party when dishonest.

On the voter side, the usage of a smart-cards changes the scheme regarding the receipt-freeness property. As already mentioned, Belenios (as well as Helios) is not receipt-free because the voter can get the random involved in the encryption and use it to prove how he voted. The fact that the encryption be performed by a smart-card prevents this, assuming the card is tamper-proof. As acknowledged in (Chaidos et al., 2016a), using such hardware tokens is indeed a way to add receipt-freeness to Belenios.

As a conclusion, BeleniosHW enjoys the same security properties as Belenios, with the addition of the receipt-freeness, based on the usage of a tamper-proof device.

5 PERFORMANCES

Exponentiation Algorithms. We hereafter define two different exponentiations that will be used to implement the algorithms. We take into account the fact that when the base is a value known in advance, it is possible to provision the card with pre-computed values.

Double Exponentiation (`DE`) is the computation of the value $u^e \cdot v^f$, when no assumption can be made on the value of u or v. Using the Straus-Shamir's trick (Straus, 1964; Gamal, 1984), this computation can be made with the same performances as computing the simple exponentiation u^e . This trick requires scanning the exponents from the most significant bits to the less significant bits, and the pre-computation of the value $u \cdot v$.

Known Value Exponentiation (`KVE`) is the computation of the value u^e, when u is a static value, known in advance (such as a group generator, or a static public key). This case can be treated as a special case of `DE` – with half-length exponents. Indeed:

$u^e = u^{e_1} \cdot v^{e_2}$, where $e = e_1 + e_2 \cdot 2^{\frac{\ell}{2}}$, $v = u^{2^{\frac{\ell}{2}}}$, and ℓ is the bit-size of e. Since u is a known value, we assume that v is available with negligible cost. As the cost of an exponentiation is linear with the exponent length, a `KVE` is twice as fast as a `DE`.

Ballot Encryption. We stick to Belenios and use the ElGamal encryption scheme. Operations are made in a group of order q, generated by a generator g. To compute $c_{\texttt{enc}}^{(i)} = \left\{\alpha^{(i)}, \beta^{(i)}\right\}$, the encryption of the choice $m^{(i)}$, we shall:

- Pick a random value $r^{(i)} \in \mathbb{Z}_q$.
- Compute $\alpha^{(i)} = g^{r^{(i)}}$, for which we can use a `KVE`.
- Compute $\beta^{(i)} = pk^{r^{(i)}} \cdot g^{m^{(i)}}$, for which we can use a `DE`.

Remark 3. *Only* β *depends on the election choice* $m^{(i)}$.

Ballot Signature. We consider that the key pair (upk_{id}, usk_{id}), generated in the register phase, has been stored in the voting card. This constitutes a slight difference with the Belenios specification (Glondu, 2018), as there is no need to compute upk_{id}. Belenios uses a Schnorr-like digital signature scheme. We denote by $\mathcal{H}_{\texttt{signature}}$ the hash function dedicated to the signature scheme. To get $s = (\textbf{challenge}, \textbf{response})$, we shall compute:

- a random value $w \in \mathbb{Z}_q$.
- $A = g^w$, using a `KVE`.
- $\textbf{challenge} = \mathcal{H}_{\texttt{signature}}(upk_{id}, A, c_{\texttt{enc}}^{(1)}, \ldots, c_{\texttt{enc}}^{(n)}) \mod q$
- $\textbf{response} = w - usk_{id} \cdot \textbf{challenge} \mod q$

Remark 4. *As A does not depend on the voter nor on the election choices, it can be pre-computed.*

Ballot Proof of Correctness. For a given ciphertext (α, β), we shall prove that it is the encryption of a valid message, i.e. that it has the form $(g^r, pk^r \cdot g^m)$, with $m \in \{M_0, \ldots, M_k\}$ ($\{0,1\}$ in our case). This is achieved with a dedicated zero-knowledge proof involving the random r used in the construction of (α, β). The proof consists in a sequence of challenges/responses, one for each possible valid choice The proof has been made non-interactive thanks to a hash function $\mathcal{H}_{\texttt{iproof}}$ to determine the sum of all challenges.

Let $m = M_s$. To create the complete proof, we perform the following steps.

Table 1: Theoretical timings for the computation of a ballot.

	DL timings (ms)				EC timings (ms)			
	# candidates				# candidates			
	1	8	16	32	1	8	16	32
Vote-indep.	240	1247	2,397	4,969	37	194	374	733
Vote-dep.	479	2493	4,795	9,397	74	383	737	1,444
Total	719	3,740	7,192	14,096	111	577	1,111	2,177

Step 1: For all $j \neq s$:

- Pick c_j, p_j at random in $\mathbb{F}_q$,
- Compute $A_j = g^{p_j} \cdot \alpha^{-c_j} = g^{p_j - r \cdot c_j}$, using a KVE,
- Compute $B_j = pk^{p_j} \cdot \left(\beta \cdot g^{-M_j}\right)^{-c_j}$, using a DE,
- Set $\textbf{challenge}_j = c_j$ and $\textbf{response}_j = p_j$.

Step 2: For the index s:

- Pick w at random in $\mathbb{F}_q$,
- Compute $A_s = g^w$ and $B_s = pk^w$,
- Set $\textbf{challenge}_s = \mathcal{H}_{\texttt{iproof}}(upk_{id}, \alpha, \beta, A_0, B_0, \ldots, A_k, B_k) - \sum_{j \neq s} \textbf{challenge}_j \mod q$
- Set $\textbf{response}_s = w + r \cdot \textbf{challenge}_s \mod q$

In our case, $m^{(i)} \in \{0,1\}$ for each ciphertext $c^{(i)}_{\texttt{enc}}$, and $\min(x) \leqslant m = \sum_i m^{(i)} \leqslant \max(x)$ for the overall proof. Hence, we have:

$$c^{(i)}_{\texttt{pr}} = \left\{ \textbf{challenge}^{(i)}_0, \textbf{response}^{(i)}_0, \textbf{challenge}^{(i)}_1, \textbf{response}^{(i)}_1 \right\}$$

$$c^{\text{overall}}_{\texttt{pr}} = \left\{ \textbf{challenge}^{\text{overall}}_{\min(x)}, \textbf{response}^{\text{overall}}_{\min(x)}, \ldots, \textbf{challenge}^{\text{overall}}_{\max(x)}, \textbf{response}^{\text{overall}}_{\max(x)} \right\}$$

Remark 5. *Note that it is possible to slightly modify the process so as to compute A_s and B_s in Step 1. Indeed, in this case $A_s = g^{p_s} \cdot \alpha^{-c_s} = g^{p_s - r \cdot c_s}$ and $B_s = pk^{p_s} \cdot \left(\beta \cdot g^{-M_s}\right)^{-c_s} = pk^{p_s} \cdot \left(pk^r g^{M_s} \cdot g^{-M_s}\right)^{-c_s} = pk^{p_s - r \cdot c_s}$. It is then sufficient to set $w = p_s - r \cdot c_s \mod q$ in the computation $\textbf{response}_s$. Doing so, all the A's can be pre-computed, and only the B's depend on the election choice. Furthermore, given an election setup, the values g^{-M_j} can be pre-computed and hence be available at no cost.*

Summary. On the voter side, a total of $3n+2$ KVE and $3n+1$ DE has to be processed. We also have noticed that among those exponentiations, $3n+2$ KVE may be processed at any time as they do not depend on the choice of the voter. For instance, the card may compute those values between two voting sessions. We can also consider pre-computing all theses values if we have access to an external storage. In this case the data would be stored encrypted with a key known only by the card, so as not to decrease the security of the solution.

The exponentiations in the encryption, as well as in the computation of the A_j's, use an ephemeral secret $r^{(i)}$. They should be implemented with a regular or an atomic algorithm to prevent side-channel attacks (see e.g. (Peeters, 2013)). We express hereafter the cost of DE and KVE in terms of "low level" operations for ℓ-bit size operands: modular multiplication $M(\ell)$, modular addition/subtraction $A(\ell)$ and modular inversion $I(\ell)$. We will uses these formulæ in the next section to evaluate the time required to compute a ballot in a voting card.

Discrete Logarithm. Let $(p, q, g, \cdot)$ denote the multiplicative group used to compute the ballot, such that $g^q \equiv 1 \mod p$. A naive implementation of DE requires 1 square and 0.75 multiplication per bit, and we assume an atomic implementation with the same timings for squares and multiplies. As already said, KVE is twice as fast. We then have the following formulæ:

$$\texttt{DL-DE}(p,q) = |q| \cdot (1.75) \cdot M(|p|)$$
$$\texttt{DL-KVE}(p,q) = 0.5 \cdot \texttt{DL-DE}(p,q)$$

Elliptic Curve. We denote by $(\mathcal{E}(p), q, g, +)$ the additive group used to compute the ballot, such that $[q]g = O$. We chose the atomic implementation proposed in (Rondepierre, 2013) with a point doubling/addition cost of 10 field multiplications and 10 field additions/subtractions and 1 doubling and 0.5 addition per bit. Each scalar multiplication also requires one modular inversion. We then have the following formulæ:

Table 2: Timings for modular operations on a dedicated hardware.

Field size (bits)	Multiplication	Addition	Inversion
		(μs)	
256	3.21	0.566	230
3072	214.05	NA	NA

$$\begin{aligned}\texttt{EC-DE}(p,q) &= |q| \cdot (1.5) \cdot \\ &\quad (10M(|p|) + 10A(|p|)) + I(|p|) \\ \texttt{EC-KVE}(p,q) &= 0.5 \cdot |q| \cdot (1.5) \cdot \\ &\quad (10M(|p|) + 10A(|p|)) + I(|p|)\end{aligned}$$

Real Component. Let us take usually recommended key lengths $|q| = 256$ and $|p| = |g| = 3072$ in the DL case and $|q| = |p| = 256$ in the EC case (in bits). In order to get an insight of the timings on an actual component, we give the figures for modular multiplication, addition/subtraction and inversion in Table 2. These figures correspond to a component equipped with a smart-card dedicated hardware efficiently implementing the modular arithmetic.

Performing ballot pre-computations saves around a third of the overall performances which is worth mentioning.

REFERENCES

Adida, B. (2008). Helios: Web-based open-audit voting. In van Oorschot, P. C., editor, *Proceedings of the 17th USENIX Security Symposium*, pages 335–348. USENIX Association.

Adida, B., de Marneffe, O., Pereira, O., and Quisquater, J.-J. (2009). Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In D. Jefferson, J.L. Hall, T. M., editor, *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. Usenix.

Belenios (2018a). Belenios election server. https://belenios.loria.fr/admin.

Belenios (2018b). Belenios verifiable online voting system. http://www.belenios.org/.

Benaloh, J., Bernhard, M., Halderman, J. A., Rivest, R. L., Ryan, P. Y. A., Stark, P. B., Teague, V., Vora, P. L., and Wallach, D. S. (2017). Public evidence from secret ballots. *CoRR*, abs/1707.08619.

Bringer, J., Chabanne, H., Kevenaar, T. A. M., and Kindarji, B. (2009). Extending match-on-card to local biometric identification. In *Biometric ID Management and Multimodal Communication, Joint COST 2101 and 2102 International Conference, BioID_MultiComm*, pages 178–186.

Budurushi, J., Neumann, S., and Volkamer, M. (2012). Smart cards in electronic voting: Lessons learned from applications in legally-binding elections and approaches proposed in scientific papers. In *EVOTE 2012*, pages 257–270.

Chaidos, P., Cortier, V., Fuchsbauer, G., and Galindo, D. (2016a). BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *Proceedings of the 2016 ACM SIGSAC*, pages 1614–1625.

Chaidos, P., Cortier, V., Fuchsbauer, G., and Galindo, D. (2016b). BeleniosRF: A Non-Interactive Receipt-Free Electronic Voting Scheme. Presentation at CCS 2016. https://www.youtube.com/watch?v=Fzj29WTVWb8.

Clarkson, M. R., Chong, S., and Myers, A. C. (2008). Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368.

Common Criteria (2018). ICs, Smart Cards and Smart Card-Related Devices and Systems , Certified Products. https://www.commoncriteriaportal.org/products/#IC.

Cortier, V., Dragan, C. C., Strub, P.-Y., Dupressoir, F., and Warinschi, B. (2018). Machine-checked proofs for electronic voting: privacy and verifiability for belenio. In *Proceedings of the 31st IEEE Computer Security Foundations Symposium (CSF'18)*. To appear.

Cortier, V., Galindo, D., Glondu, S., and Izabachène, M. (2014). Election Verifiability for Helios under Weaker Trust Assumptions. In *Computer Security - ESORICS - Proceedings, Part II*, pages 327–344.

Estonia (2018). E-Estonia. https://e-estonia.com/solutions/e-governance/i-voting/.

Gamal, T. E. (1984). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology - CRYPTO '84*, pages 10–18.

Glondu, S. (2018). Belenios specification. version 1.5. https://gforge.inria.fr/projects/belenios/.

Hao, F., Daugman, J., and Zielinski, P. (2008). A fast search algorithm for a large fuzzy database. *IEEE Trans. Information Forensics and Security*, 3(2):203–212.

Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*. Erik Learned-Miller and Andras Ferencz and Frédéric Jurie.

Lee, B. and Kim, K. (2002). Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Information Security and Cryptology - ICISC 2002, Revised Papers*, pages 389–406.

Magkos, E., Burmester, M., and Chrissikopoulos, V. (2001). Receipt-freeness in large-scale elections without untappable channels. In *Towards The E-Society: E-Commerce, E-Business, and E-Government, The First*

IFIP Conference on E-Commerce, E-Business, E-Government (I3E, pages 683–693.

Neumann, S., Feier, C., Volkamer, M., and Koenig, R. E. (2013). Towards A practical JCJ / civitas implementation. In *Informatik 2013, 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, pages 804–818.

Neumann, S. and Volkamer, M. (2012). Civitas and the real world: Problems and solutions from a practical point of view. In *Seventh International Conference on Availability, Reliability and Security, Prague, ARES*, pages 180–185.

NIST (2011). MINEX II - An Assessment of Match-on-Card technology. https://www.nist.gov/itl/iad/image-group/minex-ii-assessment-match-card-technology.

Peeters, E. (2013). *Advanced DPA Theory and Practice: Towards the Security Limits of Secure Embedded Circuits*. Springer Publishing Company, Incorporated.

Rondepierre, F. (2013). Revisiting atomic patterns for scalar multiplications on elliptic curves. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS, Revised Selected Papers*, pages 171–186.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 815–823.

Sklavos, N. and Souras, P. (2006). Economic models & approaches in information security for computer networks. *I. J. Network Security*, 2(1):14–20.

Straus, E. G. (1964). Addition chains of vectors (problem 5125). *Am. Math. Monthly*, 70:806–808.