Election Verifiability or Ballot Privacy: Do We Need to Choose?

Édouard Cuvelier, Olivier Pereira, and Thomas Peters

Université catholique de Louvain
ICTEAM – Crypto Group
1348 Louvain-la-Neuve – Belgium

Abstract. We propose a new encryption primitive, *commitment consistent encryption* (CCE), and instances of this primitive that enable building the first universally verifiable voting schemes with a perfectly private audit trail (PPAT) and practical complexity. That is:

- the audit trail that is published for verifying elections guarantees everlasting privacy, and
- the computational load required from the participants is only increased by a small constant factor compared to traditional voting schemes, and is optimal in the sense of Cramer, Gennaro and Schoenmakers [16].

These properties make it possible to introduce election verifiability in large scale elections as a pure benefit, that is, without loss of privacy compared to a non-verifiable scheme and at a similar level of efficiency.

We propose different approaches for constructing voting schemes with PPAT from CCE, as well as two efficient CCE constructions: one is tailored for elections with a small number of candidates, while the second is suitable for elections with complex ballots.

1 Introduction

Elections enable a set of voters to express their opinion regarding one or more questions, and to build an aggregate outcome from these personal opinions. While very simple elections mechanisms, like hand raising, can be very convenient to organize, various properties are usually required from voting schemes nowadays, which are not guaranteed by a hand raising process.

Vote privacy is probably the most important property that has been added on top of correctness/verifiability (guaranteed by the hand raising process), and became mandatory for public elections in most countries during the 19^{th} century, as a way to prevent coercion and bribery [36].

Elections guaranteeing the privacy of the votes while preserving the correctness of the outcome are unfortunately much harder to organize in a trustworthy way: as usual, correctness and privacy guarantees tend to conflict.

As a result, most voting schemes used today enforce privacy at the expense of the correctness properties: in traditional paper-based scheme, it is most of

J. Crampton, S. Jajodia, and K. Mayes (Eds.): ESORICS 2013, LNCS 8134, pp. 481–498, 2013.
© Springer-Verlag Berlin Heidelberg 2013

the time impossible for a voter to convince himself that his vote is included in the urns that are tallied (he has to trust election officers on that), and the same happens with the commonly deployed non verifiable electronic voting schemes, which also make it impossible for the voters to verify what is counted by the computers, if there is anything counted at all.

As a way to solve this problem, universally verifiable voting systems were proposed in the seminal works of Benaloh et al. [7, 13], works that have been followed by a considerable body of research during the last 25 years (see [12, 15, 16, 18, 22, 27, 32, 34, 35] for instance). Universally verifiable elections are realized by including in the voting process the production of an audit trail (which can be electronic, made of paper, or both) that makes it possible for voters to check that their vote was recorded properly and that the election outcome is consistent with all the votes submitted by legitimate voters (formal definitions appear in [30, 29] for instance.)

The adoption of universally verifiable technologies is however complicated if the audit trail that is provided in order to guarantee the correctness of an election in turn weakens the privacy of the votes: this raises questions about the relative importance of the correctness improvement resulting from the audit trail versus the potential decrease of privacy that results from that same audit trail, as well as about the consequences of any (even partial) failure with respect to one of these properties. These are sensitive problems, and the balance between these requirements will typically depend on the specifics of each election (stakes, voter population, culture, ...).

This compromise between correctness and privacy needs to be made in the vast majority of the verifiable voting schemes that have been proposed [7, 16, 18, 22, 27, 34, 35] (we discuss the few exceptions in Section 1.2) including those that have been used in real-world elections. The public audit trail of all those voting systems indeed includes information that could reveal individual votes if a computationally secure cryptosystem is broken, which will eventually happen in a hard to predict future, either because of the increase of power of computing devices, or because of a cryptanalytic breakthrough that can happen at any time.

For instance, Helios [3] publishes encrypted votes, which may eventually reveal those votes if the encryption scheme that is used is broken. This in part motivated the decision of the IACR to only display aliases instead of voter names on their election bulletin board: in case of broken encryption, the election bulletin board would then only reveal the content of encrypted votes but not their author (the voting server is still aware of the link between aliases and voters, though, and these aliases circulate in cleartext emails). Such a procedure however impairs eligibility verifiability, as it becomes infeasible for the voters to verify whether the ballots present on the bulletin board have been submitted by legitimate voters or are the result of ballot stuffing by the organizers [29, 4].

In a similar way, Scantegrity II [11] publishes a Q table containing the confirmation codes that have been unveiled during the voting phase, and, as soon as there are few dozen of voters, the content of this table will determine uniquely the value of the seed used to build the original P table, which in turn reveals

the votes corresponding to all voter receipts. This may be enough to defeat the purpose of the introduction of privacy in voting systems, since voters may be coerced just by fear of a future loss of privacy.

1.1 Contributions

We address this problem by proposing a new primitive, commitment consistent encryption (CCE), that can be plugged in voting schemes as a replacement for traditional encryption. The use of this primitive makes it possible to obtain verifiable elections with a perfectly private audit trail (PPAT), that is, an audit trail that preserves the privacy of the votes even when facing a computationally unbounded adversary. As a result, adding a PPAT on top of a traditional voting scheme provides the benefits of universally verifiable voting technologies without interfering with the privacy properties of the original system.

As an important example of application, we investigate the use of CCE for building single-pass [8] voting schemes with PPAT. Single-pass voting schemes support a voting process that executes asynchronously and in a single step, which makes them well-suited for large scale elections: voters just produce their ballot and send it to the authorities. The reception of the ballots and the tally are then orchestrated by a set of authorities, who are also in charge of publishing the election audit trail. The correctness of this audit trail ensures the correctness of the election outcome even if *all* authorities are corrupted. Still, the privacy of the votes relies on the number of corrupted authorities to be lower than a certain threshold.

With this application in mind, we design two efficient CCE encryption schemes. The first of our schemes is additively homomorphic and is particularly suitable for elections based on homomorphic tallying. It is however limited to elections that have a small election outcome space (e.g., elections in which the outcome is simply the sum of votes received by the candidates). Our second scheme is suitable for elections with mixnet-based tallying, in which all ballots are decrypted after shuffling, which allows supporting arbitrary ballot formats. We eventually propose a third scheme that is flexible enough to be used in both contexts but is much less efficient and complicated to use.

Our first two schemes admit simple distributed and threshold key generation procedures: all computations happen in prime order groups and the standard threshold key generation techniques available in such groups apply [24]. This is particularly important, especially in terms of round complexity, as the trustees of an election will often not be able to setup specific software for running key generation: for instance, the Helios voting system used by IACR relies on n-out-of-n distributed key generation just to keep the key generation ceremony simple (traditional threshold key generation would require more than one single round).

These two CCE schemes are also very efficient, making them usable in JavaScript applications like Helios for instance: based on the performance on the JSBN cryptographic library, the preparation of any vote that can be encoded on 256 bits requires less than a second.

Based on these schemes, we obtain the first universally verifiable voting protocols with PPAT and optimal efficiency (in the sense of [16]):

- the ballot size and the voter computational load do not depend on the number of voters nor on the number of authorities and
- the workload of the tallying authorities grows linearly with the number of voters and candidates.

Furthermore, our schemes do not rely on expensive cut-and-choose techniques: the number of exponentiations to be performed is independent of the security parameter.

1.2 Related Works

Very few voting protocols offer a perfectly private audit trail, and they all require either an amount of work by the voters that grows linearly with the number of trustees, or the use of specific communication channels, or are inefficient.

A first class of voting schemes that can offer a PPAT is based on blind signatures [22]. Here, ballots are blindly signed by an authority, then unblinded by the voters who eventually publish their authority signed ballot through an anonymous channel. The vote privacy issue is here taken care of by the anonymous channel and the audit trail only contains anonymous information. Setting up a perfectly anonymous channel can however be very challenging in a large scale election.

A second approach was proposed by Cramer, Franklin, Schoenmakers and Yung [15]. Here, a verifiable secret sharing scheme is used by the voters to distribute the information needed to tally their vote. The shares are then distributed to the authorities either through private channels or protected by encryption. The computational load of the voters then grows linearly with the number of authorities, which motivated the consecutive proposal by Cramer, Gennaro and Schoenmakers of a scheme that offers a computationally private audit trail but a work load for the voters that is independent of the number of authorities [16].

In the same spirit as the work of Cramer et al. [15], Moran and Naor proposed a voting scheme with everlasting privacy [33]. Here again, the privacy of the votes is protected through secret sharing and the complexity of the ballot preparation task grows linearly with the number of authorities.

As far as we know, our solutions are the first to offer a PPAT while being based on the third approach of e-voting, that is, the tallying of threshold encrypted ballots [7, 13, 16, 27]. In a contemporary work, Demirel, van de Graaf and Araújo [20, 19] explore a similar problem and propose a solution based on the combination of Pedersen commitments and Paillier encryption proposed of Moran and Naor [33]. As acknowledged by these authors, this solution is not practical: it relies on cut-and-choose zero-knowledge (ZK) proofs, which makes it slower than ours by approximately 4 orders of magnitude for comparable security levels, and requires the execution of sophisticated MPC protocols for distributed key generation by the trustees.

In terms of modeling, symbolic techniques also have been recently proposed to model everlasting privacy [4].

Two Flavors of Verifiability. Just like privacy, verifiability is a property that comes in computational and perfect flavors. The huge majority of schemes offer the computational variant, typically by relying on zero-knowledge proofs that only are computationally sound. The solutions we propose share this feature.

We believe that the balance between computational and perfect verifiability is however very different of the one we have for privacy. First, to have any impact, an attack on the verifiability must be mounted on-the-fly during the election: a falsified proof of verifiability proposed after 20 years will not convince anyone, while a loss of privacy after 20 years might be a practical concern. Second, the adoption of verifiable protocols is often conditioned by improvements on traditional non-verifiable systems. So, having the possibility to bring verifiability without weakening privacy (by publishing ciphertexts) might be a core decision factor. Similar considerations motivated the design of Scantegrity: its practical adoption is expected to have been facilitated by the absence of need to decrease the usability of the paper ballots [11].

Coercion resistance. The historical motivation for introducing secret ballots was the prevention of bribery or coercion. The schemes we propose address the concern of a voter who fears that the audit data of an election could reveal their vote. This concern is certainly the most ubiquitous and hard to prevent through law enforcement or by voter education: it does not require any visible step by a coercer who just needs to look at available data. We do not focus on specific coercion resistance procedures in our simple application examples, as coercion prevention is a much broader problem than what can be addressed at a protocol level, especially when vote-by-mail is authorized or when nothing prevents bringing camera phones in a voting booth. Our schemes are however compatible with most existing approaches, e.g., revoting as first used in Estonia or coercion detection [25].

Roadmap. The rest of this paper is organized as follows. Section 2 introduces our new encryption primitives, CC and CCVA encryption. Section 3 discusses security properties that these encryption primitives need to satisfy for use in voting applications. Section 4 defines two efficient CCVA schemes and explains how they can be plugged in classical voting schemes. We finally analyse the efficiency of our solutions in Section 5.

2 Commitment Consistent Encryption

We introduce a new encryption primitive, *commitment consistent encryption* (CCE). A CCE primitive is a traditional public key encryption scheme that offers an extra feature: from any CCE ciphertext, it is possible to derive a commitment on the encrypted message, and the private key can also be used to obtain an opening on that commitment. In the context of elections, we expect voters to CC encrypt their vote, which will allow authorities to compute the tally in a traditional way (e.g., by decrypting the homomorphic sum of the ciphertexts). Furthermore, when receiving a CC ciphertext, the authorities can use a

DeriveCom algorithm to derive commitments from CC ciphertexts and post that commitment on the bulletin board. This provides a PPAT if the commitments are perfectly hiding. In order to offer universal verifiability, the authorities can also make use of an Open algorithm that makes it possible to derive openings of commitments on the election tally.

For simplicity, we make our whole treatment in the single-key setting. The extension to the full threshold setting is orthogonal to our concerns and can be made using traditional techniques. In the following, an efficient algorithm runs in PPT, and a negligible function decreases faster than any inverse polynomial. An overwhelming function is close to 1 up to a negligible function.

Definition 1 (CC Encryption). *A commitment consistent encryption scheme Π is a tuple of efficient algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DeriveCom}, \mathsf{Open}, \mathsf{Verify})$ *defined as follow :*

$\mathsf{Gen}(1^n)$**:** *Given a security parameter n, output a triple (pp, pk, sk), respectively the public parameters, the public key and the secret key.*

$\mathsf{Enc}(pk, m)$**:** *Output a ciphertext c which is an encryption using the public key pk of a message m chosen in the plaintext space $\mathcal{M}$ defined by pp.*

$\mathsf{Dec}(sk, c)$**:** *From a ciphertext c, output a message m using the secret key sk.*

$\mathsf{DeriveCom}(pk, c)$**:** *Output a commitment d from a ciphertext c using pk.*

$\mathsf{Open}(sk, c)$**:** *Output an auxiliary value a using the secret key sk. This auxiliary value can be considered as part of an opening for a commitment.*

$\mathsf{Verify}(pk, d, m, a)$**:** *From a message m, a commitment d with respect to key pk and an auxiliary value a, output a bit. This algorithm checks the validity of the opening (m, a) with respect to d and pk.*

It is implicit that pp is given to each algorithm apart from Gen.

Correctness. We expect CCE schemes to satisfy the following correctness properties. For any $(pp, pk, sk) \leftarrow \mathsf{Gen}(1^n)$, any message $m \in \mathcal{M}$ and any ciphertext $c \leftarrow \mathsf{Enc}(pk, m)$, it holds with overwhelming probability in n that $\mathsf{Dec}(sk, c) = m$ and $\mathsf{Verify}(pk, \mathsf{DeriveCom}(pk, c), \mathsf{Dec}(sk, c), \mathsf{Open}(sk, c)) = 1$. For the sake of simplicity we will often shorten the expression above as $\mathsf{Verify}(pk, c)$.

The security properties that we can expect from a CCE scheme and for the derived commitments are the traditional ones and we will discuss later those that are appropriate for our applications.

The CCE definition does not guarantee that it is unfeasible to produce ciphertexts that look just like honestly computed CCE ciphertexts but are not consistent, which might be an issue for verifiable decryption. For instance, an attacker might be able to produce a ciphertext such that the DeriveCom function will provide a commitment that cannot be opened, which might be a problem if some parties are required to provide a decryption. In order to solve this problem, we introduce the concept of validity augmentation (VA) for CCE schemes.

From an operational point of view, a validity augmentation of a CCE scheme adds three algorithms: Expand, Strip and Valid. Expand augments the public key for the needs of the other algorithms. Valid takes an augmented CCE ciphertext c^{va} that contains a CCE ciphertext along with some proofs of validity, and

runs a verification procedure on those proofs to make sure that it is possible to extract from the ciphertext a commitment and an encryption of an opening for that commitment. Eventually, Strip removes those proofs to provide some homomorphic properties such as additivity on the encrypted messages.

Definition 2 (Validity augmentation). *A scheme* $\Pi^{\mathsf{VA}} := (\mathsf{VA.Gen}, \mathsf{VA.Enc}, \mathsf{VA.Dec}, \mathsf{VA.DeriveCom}, \mathsf{VA.Open}, \mathsf{VA.Verify}, \mathsf{Expand}, \mathsf{Strip}, \mathsf{Valid})$ *is a validity augmentation of the CCE scheme* $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{DeriveCom}, \mathsf{Open}, \mathsf{Verify})$ *if* Π^{VA} *is a CCE scheme equipped with three additional efficient algorithms* Expand, Strip *and* Valid *that satisfy the following conditions.*

Augmentation. VA.Gen *runs* Gen *to get* (pp, pk, sk) *and outputs an updated triple* $(pp^{\mathsf{va}}, pk^{\mathsf{va}}, sk^{\mathsf{va}}) := (pp, \mathsf{Expand}(pk), sk)$.

Validity. $\mathsf{Valid}(pk^{\mathsf{va}}, c^{\mathsf{va}}) = 1$ *for every honestly generated ciphertext and keys and, for any PPT adversary* $\mathcal{A}$*, the following probability is negligible in* n*:*

$$\Pr\left[\mathsf{Valid}(pk^{\mathsf{va}}, c^{\mathsf{va}}) = 1 \wedge \neg\mathsf{Verify}(pk, \mathsf{Strip}(pk^{\mathsf{va}}, c^{\mathsf{va}})) = 1 \mid c^{\mathsf{va}} \leftarrow \mathcal{A}(pp^{\mathsf{va}}, pk^{\mathsf{va}}); (pp^{\mathsf{va}}, pk^{\mathsf{va}}, sk^{\mathsf{va}}) \leftarrow \mathsf{VA.Gen}(1^n)\right]$$

This condition guarantees that decryption and opening succeed.

Consistency. *The distributions of* $\mathsf{Strip}(pk^{\mathsf{va}}, \mathsf{VA.Enc}(pk^{\mathsf{va}}, m))$ *and* $\mathsf{Enc}(pk, m)$ *are the same for all* m*, that is, we can strip a VA ciphertext into a normal one. Furthermore, the decryption, opening and verification of* Π^{VA} *are consistent with those of* Π*: for every ciphertext and generated keys, it must hold that* $\mathsf{VA.Dec}(sk^{\mathsf{va}}, c^{\mathsf{va}}) = \mathsf{Dec}(sk, \mathsf{Strip}(pk^{\mathsf{va}}, c^{\mathsf{va}}))$*,* $\mathsf{VA.Open}(sk^{\mathsf{va}}, c^{\mathsf{va}}) = \mathsf{Open}(sk, \mathsf{Strip}(pk^{\mathsf{va}}, c^{\mathsf{va}}))$ *and* $\mathsf{VA.Verify}(pk^{\mathsf{va}}, c^{\mathsf{va}}) = \mathsf{Verify}(pk, \mathsf{Strip}(pk^{\mathsf{va}}, c^{\mathsf{va}}))$.

We refer to the result of the augmentation of a CCE scheme as a CCVA encryption scheme or simply a CCVAE scheme.

3 Voting with a Perfectly Private Audit Trail

In the spirit of [8], we now propose a "minivoting" scheme, that we use to describe how a validity augmented CCE scheme can be used to submit ballots in an election. We then describe the security guarantees that CCE schemes need to provide for their application in voting with PPAT.

The minivoting scheme we consider follows a classic workflow. First, a setup phase takes place, during which two clean bulletin boards $\boldsymbol{PB}$ and $\boldsymbol{SB}$ are created and elections keys are generated and appropriately published. The board $\boldsymbol{PB}$ contains the public audit trail, while $\boldsymbol{SB}$ is kept secret by the authorities and used to compute the tally. Voters then produce their ballots by encrypting their votes and send these ballots to the election authorities. The ballots are processed by these authorities, and the bulletin boards are updated accordingly. At the end of the voting phase, a tallying protocol is executed and the election outcome is published.

Definition 3 (Minivoting scheme)
Let Π be a CCVA encryption scheme, and let ρ be a result function that takes a set of valid votes and produces the corresponding election outcome. From these, we build a minivoting scheme $\mathsf{Enc2Vote}(\Pi, \rho)$ *as follows.*

$\mathsf{Setup}(1^n)$ *runs the key generation algorithm* Gen *of* Π *on the same input, obtaining a triple* (pp, pk, sk)*. It also initializes a public and a secret bulletin board,* $\boldsymbol{PB}$ *and* $\boldsymbol{SB}$*, to* $\perp$*.*

$\mathsf{Vote}(pk, v)$ *is executed by voters to prepare their ballot: it encrypts a vote* v *with* pk *using* Π*, obtaining a ballot* b*.*

$\mathsf{ProcessBallot}(pk, b, \boldsymbol{PB}, \boldsymbol{SB})$ *is executed by the authorities every time a ballot is received. It rejects* b *if it is already present in* $\boldsymbol{SB}$*. Otherwise, it runs* $\mathsf{Valid}(pk, b)$ *and rejects* b *if it fails. If all these steps succeed, it appends* b *on* $\boldsymbol{SB}$ *and* $\mathsf{DeriveCom}(pk, b)$ *on* $\boldsymbol{PB}$*.*

$\mathsf{Tally}(sk, \boldsymbol{PB}, \boldsymbol{SB})$ *decrypts all ballots on* $\boldsymbol{SB}$*, obtaining a vector of votes* $\boldsymbol{v}$*, and publishes* $\rho(\boldsymbol{v})$ *on* $\boldsymbol{PB}$*.*

A minivoting scheme does not require any proof of the validity of the ballots (e.g., that they would encrypt 0 or 1 in an approval voting system), nor publishes any specific information regarding a proof of correctness of the tally, which will be needed for universal verifiability. For modularity, we address these concerns separately: the structure of these proofs of correctness will indeed be dependent of the result function ρ.

We now focus on the privacy of the votes that is offered in such a minivoting scheme, which we capture through the following experiment, slightly adapted from [8] to allow a distinction between the private and public bulletin boards.

The Vote Privacy experiment $\mathsf{VotePriv}^{\boldsymbol{B}}_{\mathcal{A},\Pi,\rho}(n)$

1. The challenger picks a bit $\beta \leftarrow \{0,1\}$ uniformly at random. He also runs the Setup algorithm of the voting scheme on input 1^n and obtains the resulting triple (pp, pk, sk) and empty bulletin boards $\boldsymbol{PB}_\beta$ and $\boldsymbol{SB}_\beta$. He then sends pp, pk to $\mathcal{A}$ and creates two other empty bulletin boards $\boldsymbol{PB}_{1-\beta}$ and $\boldsymbol{SB}_{1-\beta}$. $\mathcal{A}$ is allowed to see the board $\boldsymbol{B}_\beta$, where $\boldsymbol{B}$ is a parameter of the experiment.
2. $\mathcal{A}$ can then perform two types of queries:
   - $\mathsf{Vote}(v_0, v_1)$ On such a query, the challenger executes $\mathsf{Vote}(pk, v_i)$, obtaining a ballot b_i, and then runs $\mathsf{ProcessBallot}(pk, b_i, \boldsymbol{SB}_i)$, for $i \in \{0,1\}$.
   - $\mathsf{Ballot}(b)$ On such a query, the challenger executes $\mathsf{ProcessBallot}(pk, b, \boldsymbol{SB}_\beta)$ and, if it succeeds, also runs $\mathsf{ProcessBallot}(pk, b, \boldsymbol{SB}_{1-\beta})$.
3. The challenger computes the tally $t_0 := \mathsf{Tally}(sk, \boldsymbol{SB}_0)$ and appends t_0 on $\boldsymbol{PB}_\beta$ and $\boldsymbol{SB}_\beta$.
4. $\mathcal{A}$ outputs a bit β'. If $\beta = \beta'$ then the output of the experiment is 1 and we say that $\mathcal{A}$ wins.

Definition 4 (Perfectly Private Audit Trail). *A minivoting scheme* $\mathsf{Enc2Vote}(\Pi, \rho)$ *has a* perfectly private audit trail *(PPAT) if, for every adversary* $\mathcal{A}$*,* $\Pr[\mathsf{VotePriv}^{\boldsymbol{PB}}_{\mathcal{A},\Pi,\rho}(n) = 1] = \frac{1}{2}$.

Since this definition does not place any bound on the computational power of the adversary, the everlasting privacy of the votes is guaranteed against people who only see the $\boldsymbol{PB}$ board.

In some contexts (e.g., when using groups of unknown order), it is useful to relax the above definition by accepting statistical indistinguishability and tolerating a negligible advantage over $\frac{1}{2}$. Independently of this, the private bulletin board, only seen by the authorities, should provide computational ballot privacy.

Definition 5 (Ballot Privacy [8]). *A minivoting scheme* $\mathsf{Enc2Vote}(\Pi, \rho)$ *has* ballot privacy *if, for every PPT adversary* $\mathcal{A}$*, there is a negligible function* ϵ *such that,* $\Pr[\mathsf{VotePriv}^{\boldsymbol{SB}}_{\mathcal{A},\Pi,\rho}(n) = 1] = \frac{1}{2} + \epsilon(n)$.

Security. The following two theorems define security properties of a CCVAE scheme that guarantee the PPAT and ballot privacy of the corresponding minivoting scheme.

Theorem 1. *Let* Π *be a CCVA encryption scheme, and let* ρ *be a result function. If the output of* $\mathsf{DeriveCom}$ *is perfectly hiding, then the minivoting scheme* $\mathsf{Enc2Vote}(\Pi, \rho)$ *has a perfectly private audit trail.*

Proof. The view of the adversary is the $\mathsf{VotePriv}^{\boldsymbol{PB}}_{\mathcal{A},\Pi,\rho}$ experiment and this view is independent of β: $\boldsymbol{PB}$ only contains perfectly hiding commitments and then a tally that is always computed from $\boldsymbol{SB}_0$, which is independent of β. □

Theorem 2 ([9]). *Let* Π *be an NM-CPA CCVAE scheme, and let* ρ *be a result function. Then the minivoting scheme* $\mathsf{Enc2Vote}(\Pi, \rho)$ *has ballot privacy.*

The NM-CPA security property [21] is easy to reach from an IND-CPA encryption scheme, as shown in [9]: it is enough to augment each ciphertext with a sigma proof of knowledge of the message and randomness used to build this ciphertext. We observe that this sigma proof not only guarantees the knowledge of the plaintext and randomness, but also that the ciphertext is well-formed. We can then define a validity augmentation in a straightforward way: Expand adds the oracle $\mathcal{H}$ to the public key, Strip removes the sigma proof from the ciphertext, and Valid returns "1" only if the proof is valid. The validity condition holds thanks to the completeness and the soundness of the proof. The consistency of the augmentation is straightforward by inspection of Definition 2.

A first example of CCVAE scheme, called PPATP, based on Paillier encryption and Pedersen commitments following a suggestion by Moran and Naor [33] is available in the full version of this paper [17]. The full version also contains a generalized version of this construction with security proofs.

4 Efficient CCVAE Schemes

This section describes two efficient and usable constructions of CCVAE schemes. The first scheme, PPATS, allows using traditional ballot validity proof techniques and completing the tally through the homomorphic addition of encrypted votes.

The decryption process however involves a stage of exhaustive search of the plaintext (just as the exponential ElGamal scheme used in many applications), which restricts the use of this scheme to elections in which this kind of exhaustive search can be done, e.g., when the outcome is simply a count of the number of votes that each candidate received. The second scheme, PPATC, is tailored for mixnet based tallying procedures: the ciphertexts are not additively homomorphic but the decryption procedure is efficient regardless of the message. In both tally procedures we show explicitly how the process does not affect the PPAT as well as the ballot privacy of voting schemes provided by our CCVAE schemes.

Computational Setting. Our two efficient CCVAE schemes rely on the existence of a bilinear group generator that, on input 1^n, produces a description of bilinear groups $\Lambda_{sxdh} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, with $|q| = n$, e is an efficient and non-degenerating bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and g, h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively (we refer the reader unfamiliar with those objects to [23]). We expect that these groups are chosen in such a way that there is no known efficient mapping between $\mathbb{G}_1$ and $\mathbb{G}_2$ in either direction. This is necessary, as the security of our schemes relies on the hardness on the DDH problem in both of these groups. This setting, often called the SXDH setting, is usually considered as the choice that offers the highest level of flexibility and performance for high security parameters. Common concrete choices include the use of BLS and BN curves [5, 6].

Note that all our schemes could be adapted easily to the symmetric pairing settings, typically by relying on the hardness of the DLIN problem instead of DDH [10]. The choice we made provides more efficient protocols and also makes it possible to compute in smaller fields for equivalent security levels.

4.1 CCVA Encryption for Elections with Simple Ballots

The PPATS scheme makes use of two compatible homomorphic ingredients: ElGamal encryption and the TC2 perfectly hiding commitment scheme proposed by Abe et al. [1], which is binding in the Λ_{sxdh} setting. The resulting CCE scheme is compatible with sigma protocols, and the definition of a validity augmentation is then simple.

The PPATS *CCVAE scheme:*

$\mathsf{VA.Gen_S}(1^n)$**:** Generate $\Lambda_{sxdh} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ for $|q| = n$ together with the following additional public random generators $g_1 = g^{x_1}$ in $\mathbb{G}_1$ and $h_1 \in \mathbb{G}_2$. The triple $(pp_\mathsf{S}, pk_\mathsf{S}, sk_\mathsf{S})$ is defined as $((\Lambda_{sxdh}, h_1), g_1, x_1)$. The augmented key $pk_\mathsf{S}^{\mathsf{va}} = \mathsf{Expand}(pk_\mathsf{S})$ is computed by adding the description of an efficient hash function $\mathcal{H}$ with range $\mathbb{Z}_q$, resulting in the triple $(pp_\mathsf{S}^{\mathsf{va}} = pp_\mathsf{S}, pk_\mathsf{S}^{\mathsf{va}}, sk_\mathsf{S}^{\mathsf{va}} = sk_\mathsf{S})$.

$\mathsf{VA.Enc_S}(pk_\mathsf{S}^{\mathsf{va}}, m\,; r, s)$**:** Compute the CCE ciphertext $c = \mathsf{Enc_S}(pk_\mathsf{S}, m; r, s)$ as $(d, c_1, c_2) = (h^r h_1^m, g^s, g^r g_1^s)$ for random $r, s \in_R \mathbb{Z}_q$ and $m \in \mathbb{Z}_q$. Then compute the validity proof as follows. Compute $c' = (h^u h_1^t, g^v, g^u g_1^v)$ for random t, u, v in $\mathbb{Z}_q$. Then compute $\sigma_{cc} = (\nu_{cc}, z)$ where $\nu_{cc} = \mathcal{H}(pp_\mathsf{S}^{\mathsf{va}}, pk_\mathsf{S}^{\mathsf{va}}, c, c')$

and $z = (z_m, z_r, z_s) = (t + \nu_{cc} m, u + \nu_{cc} r, v + \nu_{cc} s)$. Output the ciphertext $c^{\mathsf{va}} = (c, \sigma_{cc})$.

$\mathsf{VA.Dec_S}(sk_{\mathsf{S}}^{\mathsf{va}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(d, c_1, c_2, \sigma_{cc})$ and return m, the discrete logarithm of $e(c_1^{x_1}/c_2, h) \cdot e(g, d)$ in basis $e(g, h_1)$.

$\mathsf{VA.DeriveCom_S}(pk_{\mathsf{S}}^{\mathsf{va}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(d, c_1, c_2, \sigma_{cc})$ and return d.

$\mathsf{VA.Open_S}(sk_{\mathsf{S}}^{\mathsf{va}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(d, c_1, c_2, \sigma_{cc})$, then compute and output the ElGamal decryption $a = c_2/c_1^{x_1}$, i.e., g^r (consisting of the TC2 auxiliary value with respect to d).

$\mathsf{VA.Verify_S}(pk_{\mathsf{S}}^{\mathsf{va}}, d, m, a)$**:** Return 1 only if $e(a, h) = e(g, d/h_1^m)$.

$\mathsf{Valid_S}(pk^{\mathsf{va}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(c, \sigma_{cc}) = (d, c_1, c_2, \nu_{cc}, z)$ and output 1 only if the proof σ_{cc} checks, that is, if $\nu_{cc} = \mathcal{H}(pp_{\mathsf{S}}^{\mathsf{va}}, pk_{\mathsf{S}}^{\mathsf{va}}, c, c')$ where $c' = \mathsf{Enc_S}(pk_S, z) \cdot c^{-\nu_{cc}}$ (with componentwise operation).

The algorithm $\mathsf{Strip_S}$ returns c from c^{va} in the obvious way. Applying $\mathsf{Strip_S}$ to PPATS ciphertexts leads to a homomorphic CCE scheme.

Theorem 3. *The* PPATS *scheme is an NM-CPA secure CCVAE scheme in the random oracle model in the Λ_{sxdh} setting.*

Proof (Sketch – See full version for details [17]). We first observe that the soundness of the Valid_S algorithm results from the one of the σ_{cc} proof, which shows that PPATS is a CCVAE scheme. The NM-CPA security of PPATS results from the observation that a PPATS ciphertext is made of a CCE ciphertext c that is IND-CPA secure, augmented with the sigma proof of knowledge of the corresponding plaintext and randomness. □

Proving vote validity. Some voting schemes require the voters to prove the validity of the votes published on $\boldsymbol{PB}$. Such proofs, which must be perfectly ZK to preserve PPAT, can be easily computed here for the Pedersen-like commitments posted on $\boldsymbol{PB}$ using standard techniques [14].

Elections with Homomorphic Tallying from PPATS. We can now use this scheme to build a voting scheme PPATSVote based on $\mathsf{Enc2Vote}(\mathsf{PPATS}, \rho_{\mathsf{S}})$ but from which we modify the Tally algorithm as follows.

1. *Stripping*: Once the polls are closed, the authorities run $\mathsf{Valid_S}$ and $\mathsf{Strip_S}$ on the CCVAE ciphertexts stored on $\boldsymbol{SB}$, obtaining CCE homomorphic ciphertexts.
2. *Aggregation*: The authorities multiply those ciphertexts, obtaining one resulting CCE ciphertext c.
3. *Decryption*: The authorities compute $v = \mathsf{Dec_S}(sk_{\mathsf{S}}, c)$ the result of the election. To prove the correctness of the decryption, they also run $\mathsf{Open_S}$ on c, obtaining an auxiliary value a. Finally the authorities append (v, a) on $\boldsymbol{PB}$.

Theorem 4. *The* PPATSVote *scheme offers a PPAT and ballot privacy in the Λ_{sxdh} setting in the random oracle model.*

Proof. The PPATSVote scheme is equivalent to the $\mathsf{Enc2Vote}(\mathsf{PPATS}, \rho_{\mathsf{S}})$ scheme except that it also discloses the auxiliary value a on $\boldsymbol{PB}$. This value is fully determined by the commitment on the outcome and by the outcome itself, which

implies that it does not provide any extra information to an unbounded adversary, and the PPAT property offered by $\mathsf{Enc2Vote}(\mathsf{PPATS}, \rho_{\mathsf{S}})$ is then preserved. The trustees having access to $\boldsymbol{SB}$ also see the decryption factors produced by Dec. They are however indistinguishable of random group elements under DDH, as for standard ElGamal decryption, and therefore do not help breaking ballot privacy. □

Audit Procedure. The audit procedure consists in the following steps:

1. Run all the verification procedures on the commitments displayed on $\boldsymbol{PB}$. If the verification procedure fails for any commitment, abort.
2. Multiply all the commitments, obtaining a commitment on the election outcome.
3. Verify that the announced outcome v and auxiliary value a are indeed an opening of the election outcome commitment. Abort if it is not the case.

The first step guarantees the validity of the votes posted, while the second and last step guarantee that the tally matches the posted votes. The binding property of the commitment scheme guarantees that the only opening that the authorities will ever be able to provide comes from a honest tallying process. We emphasize that this last verification is very efficient: it only requires the verification of an opening of one constant-size commitment—no ZK proof is needed here, contrary to traditional approaches.

As far as eligibility may be concerned, the bulletin board can also associate a name with each commitment recorded on $\boldsymbol{PB}$ without affecting the PPAT. This offers to any observers the possibility to verify that the posted votes have been submitted by valid voters (e.g., by interrogating those voters in case of doubt).

Verifiability/Accountability. Verifiability makes it possible to check whether votes have been recorded and tallied properly. In order to decide what action must be taken if a verification fails, it is sometimes useful to have a stronger property: accountability. This property was highlighted by Küsters et al. [30] and applied to the Bingo voting scheme and then to several variants of the Helios voting system [31].

While plugging the PPATS scheme into Helios would not have any noticeable impact on the verifiability analysis of Helios proposed by Kremer et al. [29], the distinction between the private and public board and between perfect and computational privacy has more impact on the accountability analyses of Küsters et al. [31]. In particular, while the ballot validity test is fully public in Helios, replacing ElGamal encryption with the PPATS scheme adds a step during which authorities could decide to reject a ciphertext because de σ_{cc} proof would be invalid, which could not be verified from the content of $\boldsymbol{PB}$ since neither σ_{cc} nor the corresponding statement appear on that board. As a result, it will not be possible to determine whether the authorities or the voter are cheating without disclosing to a judge information that only offer conditional privacy. Different strategies for improving the accountability in the case of Helios have been explored in [3, 31]. A rigorous cryptographic analysis of verifiability/accountability

of a fully-fledged voting system is an open problem (note that all current works on Helios [29, 31] abstracted the cryptographic aspects and, as result, overlooked the recently found attacks on the verifiability of Helios [9]), and is out of our scope.

4.2 CCVA Encryption for Elections with Complex Ballots

The PPATS scheme is appropriate for elections with simple ballots. In some elections, it is however useful to be able to encode complex votes in a single ciphertext. This happens for instance in elections with a very large number of candidates or with complex tallying rules that make the homomorphic aggregation approach impractical, or in elections where arbitrary write-ins need to be supported. For those elections, a tallying approach based on verifiable mixnets is usually adopted, which is the motivation for our definition of the PPATC scheme below. This scheme has an efficiency comparable to the previous one but offers efficient decryption procedures for arbitrary plaintext. The corresponding CCE scheme is however not additively homomorphic any more, but this is not a problem in a mixnet setting since ballots are individually decrypted. ElGamal encryption is a core ingredient of this scheme, together with the Λ_{sxdh}-secure and perfectly hiding commitment scheme of Abe et al [2].

The PPATC *CCVAE scheme:*

$\mathsf{VA.Gen_C}(1^n)$**:** Generate $\Lambda_{sxdh} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ for $|q| = n$ together with the following additional public rabdom generators $g_1 = g^{x_1}$, $g_2 = g^{x_2}$ in $\mathbb{G}_1$ and $h_1 \in \mathbb{G}_2$. The triple $(pp_{\mathsf{C}}, pk_{\mathsf{C}}, sk_{\mathsf{C}})$ is defined as $((\Lambda_{sxdh}, h_1), (g_1, g_2), (x_1, x_2))$. The augmented key $pk^{\mathsf{va}}_{\mathsf{C}} = \mathsf{Expand}(pk_{\mathsf{C}})$ is computed by adding to pk_{C} the description of an efficient hash function $\mathcal{H}$ with range $\mathbb{Z}_q$, resulting in the triple $(pp^{\mathsf{va}}_{\mathsf{C}} = pp_{\mathsf{C}}, pk^{\mathsf{va}}_{\mathsf{C}}, sk^{\mathsf{va}}_{\mathsf{C}} = sk_{\mathsf{C}})$.

$\mathsf{VA.Enc_C}(pk^{\mathsf{va}}_{\mathsf{C}}, m; r, r_1, r_2)$**:** Compute $c = \mathsf{Enc_C}(pk_{\mathsf{C}}, m; r, r_1, r_2)$, the CCE ciphertext $(c_1, c_2, c_3, d_1, d_2) = (g^{r_1}, g^{r_2}, g_1^r g_2^{r_2}, h^r h_1^{r_1}, m g_1^{r_1})$ for $m \in \mathbb{G}_1$ and random $r, r_1, r_2 \in_R \mathbb{Z}_q$. Then compute the following validity proof. Select random $s, s_1, s_2 \in_R \mathbb{Z}_q$ and compute the elements $c' = (c'_1, c'_2, c'_3, d'_1)$ as $(g^{s_1}, g^{s_2}, g_1^s g_2^{s_2}, h^s h_1^{s_1})$. Compute $\nu_{cc} = \mathcal{H}(pp^{\mathsf{va}}_{\mathsf{C}}, pk^{\mathsf{va}}_{\mathsf{C}}, c, c')$ and then $f = s + \nu_{cc} r$, $f_1 = s_1 + \nu_{cc} r_1$, $f_2 = s_2 + \nu_{cc} r_2$. Set $\sigma_{cc} = (\nu_{cc}, f, f_1, f_2)$. The ciphertext c^{va} is made of (c, σ_{cc}).

$\mathsf{VA.Dec_C}(sk^{\mathsf{va}}_{\mathsf{C}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$ and return $d_2 / c_1^{x_1}$.

$\mathsf{VA.DeriveCom_C}(pk^{\mathsf{va}}_{\mathsf{C}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$ and return (d_1, d_2).

$\mathsf{VA.Open_C}(sk^{\mathsf{va}}_{\mathsf{C}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \sigma_{cc})$, and return $a = c_3 / c_2^{x_2}$.

$\mathsf{VA.Verify_C}(pk^{\mathsf{va}}_{\mathsf{C}}, d_1, d_2, m, a)$**:** Return 1 if $e(g, d_1) = e(a, h) e(d_2/m, h_1)$ and 0 otherwise.

$\mathsf{Valid_C}(pk^{\mathsf{va}}, c^{\mathsf{va}})$**:** Parse c^{va} as $(c_1, c_2, c_3, d_1, d_2, \nu_{cc}, f, f_1, f_2)$ and test whether all elements of the ciphertext are properly encoded. Compute $c'_1 = g^{f_1} / c_1^{\nu_{cc}}$, $c'_2 = g^{f_2} / c_2^{\nu_{cc}}$, $c'_3 = g_1^f g_2^{f_2} / c_3^{\nu_{cc}}$ and $d'_1 = h^f h_1^{f_1} / d_1^{\nu_{cc}}$ and return 1 only if $\nu_{cc} = \mathcal{H}(pp^{\mathsf{va}}_{\mathsf{C}}, pk^{\mathsf{va}}_{\mathsf{C}}, c_1, c_2, c_3, d_1, d_2, c'_1, c'_2, c'_3, d'_1, d'_2)$.

The algorithm $\mathsf{Strip_C}$ returns c from c^{va} in the obvious way. Applying $\mathsf{Strip_C}$ to a PPATC ciphertext leads to a CCE ciphertext that is homomorphic with respect

to the curve group law in $\mathbb{G}_1$, which is sufficient for obtaining the randomization properties needed for mixing. The use of the PPATC scheme also requires the existence of an efficient mapping between the votes and $\mathbb{G}_1$. This can be realized easily in most cases. For instance, most pairing friendly curves of the form $y^2 = x^3 + b$ on $\mathbb{F}_q$ have q chosen in such a way that any message y in $\mathbb{Z}_q$ can be mapped on a point $((y^2 - b)^{\frac{1}{3}}, y)$ [6].

Theorem 5. *The* PPATC *scheme is an NM-CPA secure CCVAE scheme in the random oracle model in the Λ_{sxdh} setting.*

The proof is similar to the one of Theorem 3.

A Verifiable Shuffle for Voting Systems with PPAT. We now would like to shuffle the PPATC ciphertexts and publish openings of the corresponding anonymized commitments. Since our scheme is randomizable, this does not raise any specific concern.

We also need to make the shuffle verifiable, that is, to provide a proof of shuffle, which needs to preserve the information theoretic privacy of $\boldsymbol{PB}$. Various perfect (or statistical) ZK proof of shuffles can be used for that purpose [26, 28, 37]: these guarantee that a simulator can produce a proof of shuffle just from the inputs and output of that shuffle that is indistinguishable from a real proof, even by an unbounded adversary.

In our context, we need to verifiably shuffle, with a single permutation, both the CCE ciphertexts and the extracted commitments to keep track of their concordance. The commitment consistent shuffle approach proposed by Terelius and Wikström [38, 37] seems particularly natural for that purpose. This approach splits the proof of shuffle in two stages. First a perfectly hiding commitment on the permutation matrix used in the shuffle is computed and made public. This is the most computationally intensive part of the protocol and, interestingly, it is independent of the actual values that we need to shuffle and of the randomization factors that will be applied on the ciphertexts. Then, a much cheaper proof is produced that shows that the shuffle performed on the ciphertexts is consistent with the commitment on that permutation matrix. In our case, that proof can be computed both for the PPATC ciphertexts on $\boldsymbol{SB}$ and for the corresponding commitments on $\boldsymbol{PB}$.

We sketch the resulting tallying protocol below.

1. *Stripping*: The authorities run $\mathsf{Valid}_{\mathsf{C}}$ and $\mathsf{Strip}_{\mathsf{C}}$ on the ciphertexts stored on $\boldsymbol{SB}$, obtaining a vector $\boldsymbol{v}$ of l ciphertexts and a vector $\boldsymbol{d}$ of commitments.
2. *Permutation Commitment*: The authorities select a random permutation π and compute a commitment u on that permutation, together with a validity proof P_π.
3. *Shuffle*: The authorities select random vectors $\boldsymbol{r}, \boldsymbol{r_1}, \boldsymbol{r_2}$ from $\mathbb{Z}_q^l$ and compute a vector of ciphertexts $\boldsymbol{v'}$ where $v'_i = v_{\pi^{-1}(i)} \cdot \mathsf{Enc}_{\mathsf{C}}(pk_{\mathsf{C}}, 1, \boldsymbol{r}_{\pi^{-1}(i)}, \boldsymbol{r_1}_{\pi^{-1}(i)}, \boldsymbol{r_2}_{\pi^{-1}(i)})$ (1 represents the neutral element in $\mathbb{G}_1$). The last two components of $\boldsymbol{v'}$ are posted on $\boldsymbol{PB}$ and denoted $\boldsymbol{d'}$.

4. *Proof of shuffle*: The authorities compute two commitment consistent proofs of shuffle with respect to the committed permutation π: $P_{\boldsymbol{v}}$ that shows that $\boldsymbol{v'}$ is indeed a shuffle of $\boldsymbol{v}$ and $P_{\boldsymbol{d}}$ that shows that $\boldsymbol{d'}$ is a shuffle on $\boldsymbol{d}$. $P_{\boldsymbol{v}}$ is posted on $\boldsymbol{SB}$ and $P_{\boldsymbol{d}}$ is posted on $\boldsymbol{PB}$.
5. *Decryption of openings*: The authorities verify the proofs, then decrypt all the ciphertexts in $\boldsymbol{v'}$ and run Open on these ciphertext in order to obtain the auxiliary values for the corresponding commitment. The plaintexts and auxiliary values are published on $\boldsymbol{PB}$.

Of course, the three middle stages of this procedure, corresponding to the verifiable shuffling, should be repeated by several independent authorities.

The tally audit procedure for an observer consists in the following stages.

1. Verify the proof of permutation commitment P_π and abort if it fails.
2. Verify the proof of shuffle $P_{\boldsymbol{d}}$ and abort if it fails.
3. Verify that the authorities published valid openings for the shuffled commitments $\boldsymbol{d'}$ and abort otherwise.

The fact that this whole procedure preserves the PPAT follows from the fact that all the commitments are perfectly hiding and that all the proofs can be made perfect zero-knowledge.

5 Conclusion

We proposed a new cryptographic primitive, CCVA encryption, that enables the systematic design of voting schemes with a perfectly private audit trail. We further proposed CCVA schemes that are suitable for the organization of large-scale elections.

The PPATP scheme mentioned in section 3 and detailed in the full version of the paper [17] is fully generic and can be used with all classical tallying techniques. Its key generation algorithm is fairly sophisticated, though, and this scheme is also quite inefficient compared to our other schemes. We address then two other CCVAE schemes, PPATS and PPATC, that are much more efficient and simple to use though less flexible. They still can be used for the two most widely used vote tallying techniques: homomorphic aggregation and mixnets.

Efficiency measures. Table 1 gives an evaluation of the computational workload that our three schemes require, at comparable security levels, for computing a CCVA ciphertext and a validity proof in the case of a 0/1 vote (details appear in the full version of the paper [17].)

The first four numbers on each line count the number of exponentiations to be performed in each group – fractional values appear when non-full exponents are used. The last column, giving total costs, results from the following estimations. We associate a unit cost to the multiplication of two 256-bit integers and assume that this cost grows quadratically with the length of the operands. We target a security level equivalent to 2048-bit RSA modulus N. We select $\mathbb{G}_1$ to be taken on $\mathbb{F}_p$ for a 256-bit long prime p and $\mathbb{G}_2$ to be taken on $\mathbb{F}_{p^2}$. The cost of a

Table 1. Ciphertext computation workload

Scheme	$\mathbb{Z}_P^*$	$\mathbb{Z}_{N^2}^*$	$\mathbb{G}_1$	$\mathbb{G}_2$	Total Cost
PPATP (0/1 vote)	5.375	4	0	0	4202496
PPATP (256-bit vote)	3.375	4	0	0	3809280
PPATS (0/1 vote)	0	0	6	6	115200
PPATC (256-bit vote)	0	0	9	4	96000

point addition is evaluated to 16 multiplications in the underlying field, and the cost of a point duplication to 7 multiplications. In order to perform EC point multiplication and modular exponentiation, we consider the simple square and multiply algorithm.

As expected, this table shows very important differences between PPATP and the other two schemes: computing a PPATP ciphertext is roughly 40 times more expensive than computing a PPATC ciphertext. The cost of the PPATS and PPATC schemes is low enough to make it possible to use these schemes even on fairly slow platforms. For instance, considering the computation of a ciphertext in JavaScript in a browser using the JSBN library, which allows computing a point multiplication in a 256-bit prime order group in less than 30ms in the Chrome web browser, the computation of a PPATC ciphertext that can encode a 256-bit vote would take less than a second.

The costs of computing a PPATS and a PPATC ciphertexts are similar. The associated tallying techniques are very different though, being much more complex for PPATC. A mixnet based technique also reveals much more information than a technique based on the homomorphic aggregation of ballots. As a result, we would recommend using the PPATS scheme as long as the ballot format allows it, even if the resulting ballot preparation cost is higher than the one that would be obtained by using PPATC.

Acknowledgements. This research was supported by the WIST Walloon Region project CAMUS, the ARC French community project SCOOP and the ISEC European action grant B-CCENTRE. Édouard Cuvelier is funded by a FRIA grant of the F.R.S.-FNRS. The authors are grateful to the anonymous reviewers for their constructive feedback and they wish to thank Sylvie Baudine for her help in improving the paper.

References

1. Abe, M., Haralambiev, K., Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design. IACR Cryptology ePrint Archive 2010, 133 (2010)
2. Abe, M., Haralambiev, K., Ohkubo, M.: Group to group commitments do not shrink. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 301–317. Springer, Heidelberg (2012)
3. Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.-J.: Electing a university president using open-audit voting: Analysis of real-world use of Helios. In:

Moran, T., Jefferson, D., Hall, J.L. (eds.) Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. Usenix (August 2009)
4. Arapinis, M., Cortier, V., Kremer, S., Ryan, M.: Practical everlasting privacy. In: Basin, D., Mitchell, J.C. (eds.) POST 2013 (ETAPS 2013). LNCS, vol. 7796, pp. 21–40. Springer, Heidelberg (2013)
5. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg (2003)
6. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
7. Benaloh, J.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University (1987)
8. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting Helios for provable ballot privacy. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 335–354. Springer, Heidelberg (2011)
9. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012)
10. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
11. Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In: USENIX Security Symposium, pp. 291–306. USENIX Association (2010)
12. Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P.L.: Scantegrity: End-to-End voter-verifiable optical-scan voting. IEEE Security and Privacy 6(3), 40–46 (2008)
13. Cohen (Benaloh), J., Fischer, M.: A robust and verifiable cryptographically secure election scheme. In: Proceedings of 26th Symposium on Foundations of Computer Science, Portland, OR, pp. 372–382. IEEE (1985)
14. Cramer, R., Damgård, I., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
15. Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996)
16. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
17. Cuvelier, E., Pereira, O., Peters, T.: Election verifiability or ballot privacy: Do we need to choose? Cryptology ePrint Archive, Report 2013/216 (2013), `http://eprint.iacr.org/`
18. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
19. Demirel, D., van de Graaf, J.: A publicly-verifiable mixnet with everlasting privacy towards observers. Cryptology ePrint Archive, Report 2012/420 (2012), `http://eprint.iacr.org/`

20. Demirel, D., van de Graaf, J., Araújo, R.: Improving Helios with everlasting privacy towards the public. In: Halderman, A., Pereira, O., eds.: EVT/WOTE 2012. USENIX (2012)
21. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
22. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
23. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. 156(16) (September 2008)
24. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptology 20(1), 51–83 (2007)
25. Grewal, G.S., Ryan, M., Bursuc, S., Ryan, P.: Caveat coercitor: Coercion-evidence in electronic voting. In: IEEE Security and Privacy Symposium (2013)
26. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Journal of Cryptology 23(4), 546–579 (2010)
27. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
28. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: USENIX Security Symposium, pp. 339–353. USENIX (2002)
29. Kremer, S., Ryan, M., Smyth, B.: Election verifiability in electronic voting protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 389–404. Springer, Heidelberg (2010)
30. Küsters, R., Truderung, T., Vogt, A.: Accountability: definition and relationship to verifiability. In: CCS 2010, pp. 526–535. ACM (2010)
31. Küsters, R., Truderung, T., Vogt, A.: Clash attacks on the verifiability of e-voting systems. In: IEEE Symposium on Security and Privacy, S&P 2012, pp. 395–409. IEEE Computer Society (2012)
32. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
33. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. ACM Trans. Inf. Syst. Secur. 13(2) (2010)
34. Ryan, P.Y.A., Bismark, D., Heather, J., Schneider, S., Xia, Z.: The Prêt à Voter verifiable election system. IEEE Transactions on Information Forensics and Security 4, 662–673 (2009)
35. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
36. Saltman, R.G.: The history and politics of voting technology. Palgrave Macmillan (2006)
37. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 100–113. Springer, Heidelberg (2010)
38. Wikström, D.: A commitment-consistent proof of a shuffle. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 407–421. Springer, Heidelberg (2009)