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 among those properties, and became mandatory for public elections in most countries during the 19^{th} century, as a way to prevent coercion and bribery [34]. 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 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, 15–17, 20, 24, 29, 32, 33] 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 [28, 26] 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, 17, 20, 24, 32, 33] (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 [26, 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 [21]. 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 [20]. 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 [30]. 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, 24]. In a contemporary work, Demirel, van de Graaf and Araújo [18, 19] explore a similar problem and propose a solution based on the combination of Pedersen commitments and Paillier encryption proposed of Moran and Naor [30]. 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 comes in computational and information theoretic flavors, election verifiability can be computational or information theoretic. Again, just as in the case ballot privacy, the verifiability property is computational in most of the efficient voting schemes known today. A common place where this computational aspects appears is in the zero-knowledge proofs that are used in these schemes, which are usually only computationally sound (typically relying on the Fiat-Shamir heuristic). If computational assumptions are broken, this could allow voters to fake a vote validity proof, or trustees to fake a decryption proof. The lack of soundness of these proofs might become apparent in an unpredictable future, when people will become able to break the encryption scheme that is used, but this is expected to happen way too late to provide an effective way of correcting the election outcome.

We point important practical differences between the effects of computational privacy and computational verifiability.

1. While breaking the privacy of the votes can be harmful at any time, an adversary who would like to fake the outcome and the audit trail of an election needs to break the system during the election (that is, provide audit information that would pass all verification procedures even though the properties that the verification is supposed to guarantee would be violated), since this is the time when the audit trail must be published. It seems much more demanding to be able to break a scheme in real time than to break it in some future.

2. Universal verifiability is a correctness property that people adopt by comparing verifiable systems with the non-verifiable systems that they used in the past. We believe that this adoption can be simplified if it does not impact the other properties of the system and, in particular, if the audit trail that is produced does not decrease the privacy properties of the previously used systems. (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].)

We believe that those reasons support the development of voting schemes with PPAT.

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 [22].

Roadmap. The rest of this paper is organized as follows. Section 2 introduces the tools and the computational assumptions we use. In Section 3, we introduce our new encryption primitives, CC and CCVA encryption. Section 4 discusses security properties that these encryption schemes need to satisfy for use in voting applications. Then, Section 5 describes a generic construction of CC and CCVA encryption schemes. Section 6 defines two efficient CCVA encryption schemes and explains how they can be plugged in classical voting schemes. We finally analyze the efficiency of our solutions in Section 7.

2 Preliminaries

An *efficient* adversary is one whose running time is bounded by a polynomial in the length of its input. A function $\epsilon : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is *negligible* if for any $c \in \mathbb{N}$ we have that $\lim_{n\to\infty} \epsilon(n) \cdot n^c = 0$. An *overwhelming* function is close to 1 up to a negligible function. Finaly, $|n|$ is the bit-length of the positive integer n.

Building blocks We review some standard definitions and introduce the corresponding notations.

Definition 1 (Encryption Scheme). *An encryption scheme Π_{E} consists of a triple of efficient algorithms $(\mathsf{Gen}_{\mathsf{E}}, \mathsf{Enc}_{\mathsf{E}}, \mathsf{Dec}_{\mathsf{E}})$ such that:*

$\mathsf{Gen}_{\mathsf{E}}(1^n)$**:** *Output a pair of public and secret keys (pk, sk). The public key specifies a message space $\mathcal{M}_E$.*

$\mathsf{Enc}_{\mathsf{E}}(pk, m)$**:** *Given a public key and a message m, return $\perp$ if $m \notin \mathcal{M}_E$, otherwise output a ciphertext c. In some cases, we explicit the notation $\mathsf{Enc}_{\mathsf{E}}(pk, m; r)$ for the randomness used in the encryption.*

$\mathsf{Dec}_{\mathsf{E}}(sk, c)$**:** *Using the private key sk, output a decryption d of the ciphertext c.*

These algorithms must satisfy the following correctness condition. If (pk, sk) is sampled from $\mathsf{Gen}_{\mathsf{E}}(1^n)$ and c is the outcome of $\mathsf{Enc}_{\mathsf{E}}(pk, m)$ with $m \in \mathcal{M}_E$, then $\mathsf{Dec}_{\mathsf{E}}(sk, c) = m$ with overwhelming probability.

Definition 2 (Commitment Scheme). *A commitment scheme Π_C is a triple of efficient algorithms $(\mathsf{GenC}, \mathsf{Com}, \mathsf{Verify}_{\mathsf{C}})$ such that :*

$\mathsf{GenC}(1^n)$**:** *Output a public commitment key ck. The public key specifies a message space $\mathcal{M}_C$ and a auxiliary value space $\mathcal{M}_{aux}$.*

$\mathsf{Com}(ck, m)$**:** *Output a pair (d, a) made of a commitment and an auxiliary value.*

$\mathsf{Verify}_{\mathsf{C}}(ck, d, a, m)$**:** *Return either 1 or 0 if these inputs are consistent or not.*

These algorithms must satisfy the following correctness condition. If ck is sampled from $\mathsf{GenC}(1^n)$, (d, a) is the outcome of $\mathsf{Com}(ck, m)$ with $m \in \mathcal{M}_C$, then $\mathsf{Verify}_{\mathsf{C}}(ck, d, a, m) = 1$ with overwhelming probability.

Definition 3 (Sigma Protocol).
A sigma protocol for an NP language $\mathcal{L} : \{s \mid \exists w : L(s,w) = 1\}$ *is a pair of interactive algorithms* (P, V) *that work as follows. On input* (s, w) *for* P *and* s *for* V*, the following three moves interaction takes place:*

1. *P outputs a "commitment" comm to the verifier.*
2. *V selects a challenge chal uniformly at random from a challenge space and sends it to the prover.*
3. *P sends a "response" resp and halts.*

Eventually, V evaluates a predicate Verify *on the statement s and the transcript* $(comm, chal, resp)$ *and returns* 0 *or* 1*, then halts.*

The proofs that we use satisfy the following properties, which are satisfied by most traditional sigma protocols.

Completeness An honest run between $P(s,w)$ and $V(s)$ always accepts if $L(s,w) = 1$

Special soundness There exists an extractor that, on input of two valid transcripts (t_1, t_2, t_3) and (t_1, t'_2, t'_3) w.r.t. s, returns the correct witness w. As a result, it is hard to produce invalid proofs.

Special honest verifier ZK It is possible to simulate any proof made with a honest verifier and it is possible to do so for any given challenge.

Perfect ZK The simulated proofs are distributed just as honest proofs.

Unique response For any statement, commitment and challenge, there exists at most one response that would lead to an accepting transcript.

More detailed definitions, as well as techniques for making these proofs non-interactive are available in [9] for instance.

Computational setting

We rely on the Decisional Composite Residuosity [31] problem for the security of our Paillier based scheme in Section 5.

Assumption 1 (DCR) *Let n and $l(n)$ be security parameters. Let $N = pq$ where $p = 2p'+1, q = 2q'+1$, and $p' \neq q'$ are uniformly distributed over l-bit numbers such that p, q, p', q' are primes. It is hard to distinguish random elements of $\mathbb{Z}^*_{N^2}$ from random elements of the subgroup of N-th powers of elements of $\mathbb{Z}^*_{N^2}$.*

The security of our two efficient schemes described in Section 6 only relies on the hardness of the Decisional Diffie-Hellman problem.

Assumption 2 (DDH) *Let n be a security parameter. Let g be a generator of a group $\mathbb{G}$ of prime order q such that $|q| = n$. It is hard to distinguish the tuple (g^a, g^b, g^{ab}) from the tuple (g^a, g^b, g^c) where $a, b, c \in \mathbb{Z}_q$ are chosen uniformly at random.*

We 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 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 will rely 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.

3 Commitment Consistent Encryption

We introduce a new encryption primitive, *commitment consistent encryption* (CCE). A CCE scheme 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, voters will be expected 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), while only posting the derived commitments on the bulletin board, thus providing a PPAT if the commitments are perfectly hiding.

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.

Definition 4 (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 all these algorithms.

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.

As a first example of CCE motivating our further developments, consider the following El-Gamal based CC encryption scheme:

$\mathsf{Gen}(1^n)$: Output a triple (pp, pk, sk) where pp is the description of a group G of prime order q. The public key pk consists of two generators g and h of G as well as an efficient hash function $\mathcal{H}$ while the secret key sk is α where $g^\alpha = h$.

$\mathsf{Enc}(pk, m)$: For a message m, select randomly $r, s \in \mathbb{Z}_q$, compute $c_0 = g^r$, $d = g^m h^r$, $e = \mathcal{H}(pp, c_0, h^r, g^s, h^s)$ and $z = er + s$ and output $c = (c_0, d, e, z)$.

$\mathsf{Dec}(sk, c)$: Parse c as (c_0, d, e, z) and output m as the result of the computation of the discrete logarithm of d/c_0^α in basis g.

$\mathsf{DeriveCom}(pk, c)$: Parse c as (c_0, d, e, z) and output d.

$\mathsf{Open}(sk, c)$: Parse c as (c_0, d, e, z) and output $a = (c_0, a_0, e, z)$ where $a_0 = c_0^\alpha$.

$\mathsf{Verify}(pk, m, d, a)$: Parse a as above and test if $e \stackrel{?}{=} \mathcal{H}(pp, c_0, a_0, g^z c_0^{-e}, h^z a_0^{-e})$.

This encryption scheme produces traditional ElGamal ciphertexts together with a ZK proof of knowledge of the randomness and plaintext. The second element of the ElGamal ciphertext can be seen as a Pedersen commitment, which is made binding thanks to the ZK proof. While offering a particularly simple verifiable decryption procedure (only a_0 needs to be computed), this scheme has two major limitations for the applications we have in mind: it is not homomorphic, and the validity of ciphertexts cannot be determined without

decryption. The first limitation prevents using traditional tallying procedures, while the second could make a tallying procedure fail if encrypted votes happen to be invalid.

We address these two concerns by introducing a notion of validity augmentation (VA) for CCE schemes. A VA guarantee the validity of CCE ciphertext. In the context of an election, this makes sure that the authorities are able to produce a tally from the ciphertexts they receive.

Contrary to CCE ciphertexts, our VA ciphertexts do not need to be homomorphic: as soon as the authorities are convinced of the validity of a VA ciphertext, they can strip it and recover a homomorphic CCE ciphertext for tallying.

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 and runs a verification procedure on that ciphertext to make sure that it is possible to extract from it a commitment and an encryption of an opening for that commitment. This is the crucial part to convince the authorities that they will indeed be able to produce a tally. Eventually, Strip removes those proofs to provide some homomorphic properties.

Definition 5 (Validity augmentation). *A scheme* $\Pi^{\mathsf{VA}} := (\mathsf{VA.Gen}, \mathsf{VA.Enc}, \mathsf{VA.Dec}, \mathsf{VA.DeriveCom}, \mathsf{VA.Open}, \mathsf{VA.Verify})$ *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. $\mathsf{VA.Gen}$ *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*:* $\Pr[\mathsf{Valid}(pk^{\mathsf{va}}, c^{\mathsf{va}}) = 1 \wedge \neg\mathsf{Verify}(pk, \mathsf{Strip}(c^{\mathsf{va}})) = 1 : c^{\mathsf{va}} \leftarrow \mathcal{A}(pp^{\mathsf{va}}, pk^{\mathsf{va}}); (pp^{\mathsf{va}}, pk^{\mathsf{va}}, sk^{\mathsf{va}}) \leftarrow \mathsf{VA.Gen}(1^n)]$*. 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* $\mathsf{VA.Dec}(sk^{\mathsf{va}}, c) = \mathsf{Dec}(sk, \mathsf{Strip}(pk^{\mathsf{va}}, c))$, $\mathsf{VA.Open}(sk^{\mathsf{va}}, c) = \mathsf{Open}(sk, \mathsf{Strip}(pk^{\mathsf{va}}, c))$ *and finally* $\mathsf{VA.Verify}(pk^{\mathsf{va}}, c) = \mathsf{Verify}(pk, \mathsf{Strip}(pk^{\mathsf{va}}, c))$.

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

According to this definition, validity augmented schemes are validity augmented for themselves. While this is not a problem, it will not be useful since the purpose of the augmentation is to provide a validity verification mechanism while not being required to preserve the homomorphic properties of the basic scheme.

4 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 6 (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* $\perp$*.*

$\mathsf{Vote}(pk, v)$ *encrypts a vote* v *with* pk *using* Π*, obtaining a ballot* b*.*

$\mathsf{ProcessBallot}(pk, b, \boldsymbol{SB})$ *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* $\boldsymbol{PB}$*.*

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

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.

This experiment is the basis of our definition of a PPAT.

Definition 7 (Perfectly Private Audit Trail). *A minivoting scheme* $\mathsf{Enc2Vote}(\Pi, \rho)$ *has a* perfectly private audit trail *(PPAT) if, for every adversary* $\mathcal{A}$*,* $\Pr[\mathsf{VotePriv}^{\boldsymbol{PB}}_{\mathcal{A},\Pi,\rho}(n) = 1] = \frac{1}{2}$.

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 8 (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.*

5 Generic Construction of a CCVAE Scheme

We now propose a generic construction of a CCVAE scheme that is suitable for voting with PPAT, as well as a simple instance of this construction based on a combination of the Paillier cryptosystem and a Pedersen commitment.

5.1 Building a CCVAE Scheme from Standard Building Blocks

Our generic construction of a CCE scheme is based on a traditional homomorphic commitment scheme that is made of a key generation algorithm GenC, a commitment algorithm Com that provides a commitment and the opening auxiliary value, and an opening verification algorithm Verify that allows checking the validity of an opening. Our construction also relies on two homomorphic encryption schemes for encrypting the committed message and the auxiliary value.

No specific tallying approach is discussed for now, even though most standard approaches could be used. We delay this aspect of the protocols to the more efficient constructions that come next.

The Com + Enc$_1$ + Enc$_2$ construction Let $\Pi_{\mathsf{C}} = (\mathsf{GenC}, \mathsf{Com}, \mathsf{Verify}_{\mathsf{C}})$ be a commitment scheme and $\Pi_{\mathsf{E}}^i = \left(\mathsf{Gen}_{\mathsf{E}}^i, \mathsf{Enc}_{\mathsf{E}}^i, \mathsf{Dec}_{\mathsf{E}}^i\right)$ be an encryption scheme for $i = 1, 2$. We assume the following mild condition for the generation algorithms: GenC, $\mathsf{Gen}_{\mathsf{E}}^1$ and $\mathsf{Gen}_{\mathsf{E}}^2$ can be run on a common input returned by a probabilistic algorithm Setup such that we will get $\mathcal{M}_{\mathsf{E}}^1 \subset \mathcal{M}_{\mathsf{C}}$ and $\mathcal{M}_{\mathsf{aux}} \subset \mathcal{M}_{\mathsf{E}}^2$ (using the notations of Section 2.) Then we define the following commitment consistent encryption scheme $\Pi_{\mathsf{G}} = (\mathsf{Gen}_{\mathsf{G}}, \mathsf{Enc}_{\mathsf{G}}, \mathsf{Dec}_{\mathsf{G}}, \mathsf{DeriveCom}_{\mathsf{G}}, \mathsf{Open}_{\mathsf{G}}, \mathsf{Verify}_{\mathsf{G}})$.

$\mathsf{Gen}_{\mathsf{G}}(1^n)$**:** Run $\mathsf{Setup}(1^n)$ to get a common public parameter pp and compute $ck \leftarrow \mathsf{GenC}(pp)$, $(pk_1, sk_1) \leftarrow \mathsf{Gen}_{\mathsf{E}}^1(pp)$ and $(pk_2, sk_2) \leftarrow \mathsf{Gen}_{\mathsf{E}}^2(pp)$. Output $pk = (ck, pk_1, pk_2)$ and $sk = (sk_1, sk_2)$ with the above specification.

$\mathsf{Enc}_{\mathsf{G}}(pk, m)$**:** Parse pk as (ck, pk_1, pk_2). For $m \in \mathcal{M} \subset \mathcal{M}_{\mathsf{E}}^1$ compute $(com, a) \leftarrow \mathsf{Com}(ck, m)$, $enc_1 \leftarrow \mathsf{Enc}_{\mathsf{E}}^1(pk_1, m)$ and $enc_2 \leftarrow \mathsf{Enc}_{\mathsf{E}}^2(pk_2, a)$. Output the ciphertext $c = (com, enc_1, enc_2)$.

$\mathsf{Dec}_{\mathsf{G}}(sk, c)$**:** Parse sk as (sk_1, sk_2) and c as (com, enc_1, enc_2) and return $\perp$ if it has not the right form. Otherwise return the plaintext $m = \mathsf{Dec}_{\mathsf{E}}^1(sk_1, enc_1)$.

$\mathsf{DeriveCom}_{\mathsf{G}}(pk, c)$**:** Parse pk as (ck, pk_1, pk_2) and c as (com, enc_1, enc_2) and return $\perp$ if it has not the right form. Otherwise return $d = com$.

$\mathsf{Open}_{\mathsf{G}}(sk, c)$**:** Parse sk as (sk_1, sk_2) and c as (com, enc_1, enc_2) and return $\perp$ if it has not the right form. Otherwise return the value $a = \mathsf{Dec}_{\mathsf{E}}^2(sk_2, enc_2)$.

$\mathsf{Verify}_{\mathsf{G}}(pk, d, m, a)$**:** Parse pk as (ck, pk_1, pk_2) and output $\mathsf{Verify}_C(ck, d, m, a)$.

The following theorem shows that schemes built according to this approach, using secure components, keep the same level of security.

Theorem 3. *Let Π_{C} be a computationally hiding commitment scheme and Π_{E}^1, Π_{E}^2 be two IND-CPA secure encryption schemes which together support the* **Com +Enc$_1$ + Enc$_2$** *construction. Then, the resulting scheme Π_{G} consists in an IND-CPA secure CCE scheme.*

Proof. First the CCE correctness follows immediately from correctness of the underlying components. Now, let us focus on the security property. Let $\mathcal{A}$ be an adversary against Π_{G}. From $\mathcal{A}$ we build an adversary $\mathcal{A}'$ against at least one of the underlying schemes Π_{C}, Π_{E}^1 or Π_{E}^2, which succeeds with a closely related success probability. For convenience we denote Π_{C} by Π^0. Let C^0 be the space of commitments, C^i the spaces of ciphertexts for $i = 1, 2$ and $\mathsf{C}_{\mathcal{G}}$ the space of ciphertexts for Π_{G}. Likewise, $\mathcal{M}^i$ denotes the message space for Π^i for $i = 0, 1, 2$ and $\mathcal{M}$ is the plaintext space for Π_{G} .

Given any message $m \in \mathcal{M}$, we consider the following distributions.

$\mathcal{D}_3$ is the distribution $(com, enc_1, enc_2) \leftarrow \mathsf{Enc}_{\mathsf{G}}(m)$ of the encryptions of m in Π_{G}.

$\mathcal{D}_2$ Defined as $\mathcal{D}_3$ except for the generation of enc_2. Instead it computes $enc_2^* \leftarrow \mathsf{Enc}_{\mathsf{E}}^2(a^*)$ for a random auxiliary value a^*.

$\mathcal{D}_1$ Same as $\mathcal{D}_2$ except that enc_1 is replaced by $enc_1^* \leftarrow \mathsf{Enc}_{\mathsf{E}}^1(m^*)$ for a random m^*.

$\mathcal{D}_0$ Same as $\mathcal{D}_1$ except that com is replaced by com^* computed with an independent random value.

Remark that $\mathcal{D}_0$ is the uniform random distribution on $\mathsf{C}^0 \times \mathsf{C}^1 \times \mathsf{C}^2$ which is not the uniform random distribution on $\mathsf{C}_{\mathcal{G}}$.

In the CPA game, $\mathcal{A}$ outputs (m_0, m_1) and must distinguish $\mathcal{D}_3(m_0)$ from $\mathcal{D}_3(m_1)$. We define

$$\mathrm{Adv}_{\mathcal{A}}^{\Pi_{\mathsf{G}}}(n) = |\Pr[1 \leftarrow \mathcal{A}(\mathcal{D}_3(m_0))] - \Pr[1 \leftarrow \mathcal{A}(\mathcal{D}_3(m_1))]|$$

and we set $P_{b,k} = \Pr[1 \leftarrow \mathcal{A}(c) | c \leftarrow \mathcal{D}_k(m_b)]$.

We bound $\mathrm{Adv}_{\mathcal{A}}^{\Pi_{\mathsf{G}}}(n)$ by a linear combination of $\mathrm{Adv}_{\mathcal{A}'}^{\Pi_k}(n)$. Note that the CPA game for $\mathcal{A}'$ is defined for a slightly different but equivalent experiment : $\mathcal{A}'$ has to distinguish whether an output in C^k is computed from a chosen message or from a uniformly distributed message in $\mathcal{M}^k$.

Let us see how $\mathcal{A}'$ works. $\mathcal{A}'$ runs $\mathsf{Gen}(1^n)$ generating instances for Π^0, Π^1 and Π^2. On these inputs, A' runs $\mathcal{A}$ and receives (m_0, m_1). $\mathcal{A}'$ flips two coins $b \leftarrow 0, 1$ and $k \leftarrow 0, 1, 2$. Depending on b and k, $\mathcal{A}'$ follows one of the next patterns.

- $k = 0$ $\mathcal{A}'$ computes enc_1^* and enc_2^* identically to $\mathcal{D}_0(m_b)$ and $\mathcal{D}_1(m_b)$. Then $\mathcal{A}'$ queries the Π^0 oracle $\mathcal{O}^0$ on m_b and receives com'. $\mathcal{O}^0$ computes com' as $\mathsf{Com}(m_b)$ or as $\mathsf{Com}(m^*)$ by tossing a coin. Finally $\mathcal{A}'$ sets $c = (com', enc_1^*, enc_2^*)$.
- $k = 1$ $\mathcal{A}'$ computes com and enc_2^* identically to $\mathcal{D}_1(m_b)$ and $\mathcal{D}_2(m_b)$. Then $\mathcal{A}'$ queries the Π^1 oracle $\mathcal{O}^1$ on m_b and receives enc_1'. $\mathcal{O}^1$ computes enc_1' as $\mathsf{Enc}_{\mathsf{E}}^1(m_b)$ or as $\mathsf{Enc}_{\mathsf{E}}^1(m^*)$ by tossing a coin. Finally $\mathcal{A}'$ sets $c = (com, enc_1', enc_2^*)$.
- $k = 2$ $\mathcal{A}'$ computes com and enc_1 identically to $\mathcal{D}_2(m_b)$ and $\mathcal{D}_3(m_b)$. Then $\mathcal{A}'$ queries the Π^2 oracle $\mathcal{O}^2$ on the auxiliary value a given by the computation of the commitment and receives enc_2'. $\mathcal{O}^2$ computes enc_2' as $\mathsf{Enc}_{\mathsf{E}}^2(a)$ or as $\mathsf{Enc}_{\mathsf{E}}^2(a^*)$ by tossing a coin and sends enc_2' to $\mathcal{A}'$. Finally $\mathcal{A}'$ sets $c = (com, enc_1, enc_2')$.

In all these cases, $\mathcal{A}'$ sends c to $\mathcal{A}$. $\mathcal{A}'$ outputs the guess of $\mathcal{A}$. It is clear that $\mathcal{A}'$ runs in polynomial time if $\mathcal{A}$ does. Using the triangular inequality, from $\mathrm{Adv}_{\mathcal{A}}^{\Pi_{\mathsf{G}}}(n) = |P_{0,3}(n) - P_{1,3}(n)|$ and $P_{0,0}(n) = P_{1,0}(n)$, we have:

$$\begin{aligned}
\mathrm{Adv}_{\mathcal{A}}^{\Pi_{\mathsf{G}}}(n) &\leq \sum_{k=0}^{2}\sum_{b=0}^{1} |P_{b,k}(n) - P_{b,k+1}(n)| \\
&\leq \sum_{k=0}^{2} |P_{0,k+1}(n) - P_{0,k}(n)| + |P_{0,0}(n) - P_{1,0}(n)| \\
&\quad + \sum_{k=0}^{2} |P_{1,k}(n) - P_{1,k+1}(n)| \\
&\leq \frac{1}{6}\sum_{k=0}^{2}\sum_{b=0}^{1} \mathrm{Adv}_{\mathcal{A}'}^{\Pi_k}(n) \;\leq\; negl(n)
\end{aligned}$$

This concludes the proof. □

Validity augmentation If the commitment and encryption schemes used in the construction above are compatible with sigma proofs, then it is easy to augment this IND-CPA secure CCE scheme into an NM-CPA secure CCVAE scheme, hence obtaining the conditions needed for Theorem 2 to apply.

Theorem 4 ([9], informal). *Let Π_{E} be an IND-CPA secure encryption scheme and (P, V) be a sigma proof of knowledge for the language $\mathsf{L}((pk, c), (m, r)) = 1 \Leftrightarrow c = \mathsf{Enc}_{\mathsf{E}}(pk, m; r)$ with special soundness, special honest verifier zero-knowledge, unique responses and a challenge space $\mathcal{R}$, for range, whose inverse of the size is negligible. Let $\mathcal{H}$ be a random oracle $\{0,1\}^* \to \mathcal{R}$. Then the following scheme is NM-CPA secure:*

- *The key generation algorithm is the one of Π_{E} augmented with $\mathcal{H}$.*

– *In order to encrypt m, run the encryption algorithm of Π_{E}, then compute the corresponding proof of knowledge made non-interactive using the Fiat-Shamir transform.*
– *In order to decrypt a ciphertext, check the proof and in case of validity run the decryption algorithm of Π_{E} on the ciphertext part, or return $\perp$ otherwise.*

This sigma proof immediately provides a validity augmentation. More precisely, in the case of our Π_{G} construction, we obtain a proof for the language $\mathsf{L}((pp, pk, c), (m, a, r_c, r_1, r_2)) = 1 \Leftrightarrow c = (com, enc_1, enc_2)$, $(com, a) = \mathsf{Com}_1(m, r_c)$, $enc_1 = \mathsf{Enc}_1(m, r_1)$ and $enc_2 = \mathsf{Enc}_2(a, r_2)$: such a proof not only guarantees the knowledge of the plaintext and randomness, but also that the ciphertext is valid.

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 5.

Eventually, if the commitment scheme in the generic construction is perfectly-hiding, then all the conditions of Theorem 1 and Theorem 2 hold: the resulting mini-voting scheme enjoys the PPAT as well as the ballot-privacy.

Adding extra proofs. The minivoting scheme resulting from Π_{G} allows any voter to check whether her/his vote is posted on the public bulletin board. Depending on the election specifics, extra proofs can be added in order to prove the validity of the votes, or to prove that the tally is consistent with the posted commitments. These proofs will not influence the PPAT property as long as they are perfect (or at least statistical) zero-knowledge.

5.2 Instance based on Pedersen and Paillier

The PPATP scheme. The Π_{G} construction can be instantiated using a combination of Paillier and Pedersen commitments, inspired from a proposal by Moran and Naor [30]. This leads to a scheme that we call PPATP: the idea is to select a traditional Paillier modulus N^2, and then to perform the Pedersen commitments in a subgroup of the quadratic residues modulo a prime $P = 2kN + 1$, whose order equals N. The small public odd k-value is not fixed to facilitate the generation of P.

The underlying CCE scheme of PPATP*:*

$\mathsf{Gen}_{\mathsf{P}}(1^n)$**:** Choose two n-bit safe primes, $p = 2p'+1$ and $q = 2q'+1$, and compute the public modulus $N = pq$ for the Paillier encryption. The corresponding secret key $\lambda := \lambda(N)$ is the number $\mathrm{lcm}(\phi(p), \phi(q)) = 2p'q' = \phi(N)/2$. Then pick a prime $P = 2kN + 1$ and two public generators G and H of a N-order subgroup of the quadratic residues QR_P for the Pedersen-like commitment. Output $pp_{\mathsf{P}} = (P, N)$, $pk_{\mathsf{P}} = (G, H)$ and $sk_{\mathsf{P}} = \lambda$.

$\mathsf{Enc}_{\mathsf{P}}(pk_{\mathsf{P}}, m\,; r, s, t)$**:** For $m \in \mathbb{Z}_N$ and for random $r \in_R \mathbb{Z}_N$ and $s, t \in_R \mathbb{Z}_N^*$ compute $d = G^m H^a$ in $\mathbb{Z}_P^*$ and, $c_1 = (N+1)^m s^N$ and $c_2 = (N+1)^r t^N$ in $\mathbb{Z}_{N^2}^*$. Output the ciphertext $c = (d, c_1, c_2)$.

$\mathsf{Dec}_{\mathsf{P}}(sk_{\mathsf{P}}, c)$**:** Parse c as (d, c_1, c_2) and compute $m_0 = ([c_1^{\lambda} \bmod N^2] - 1)/N$. Output $m = m_0 \cdot [\lambda^{-1} \bmod N]$ in $\mathbb{Z}_N$ as in the Paillier decryption.

$\mathsf{DeriveCom}_{\mathsf{P}}(pk_{\mathsf{P}}, c)$**:** Parse c as (d, c_1, c_2) and output d.

$\mathsf{Open}_{\mathsf{P}}(sk_{\mathsf{P}}, c)$**:** Parse c as (d, c_1, c_2) and compute $a_0 = ([c_2^{\lambda} \bmod N^2] - 1)/N$. Output $a = a_0 \cdot [\lambda^{-1} \bmod N]$ in $\mathbb{Z}_N$ as in the Paillier decryption.

$\mathsf{Verify}_{\mathsf{P}}(pk, d, m, a)$**:** Output 1 if $d = G^m H^a \bmod P$. Otherwise output 0.

Thanks to the careful choice of the parameters (the messages and auxiliary values lie in $\mathbb{Z}_N$), the scheme Π_{P} is additively homomorphic, which makes it convenient for most usual tallying techniques. This homomorphic property also makes it possible to apply the generic transformation described above by using a simple non interactive sigma protocol that enjoys all the properties we need for the validity augmentation.

The validity augmentation of PPATP*:*

$\mathsf{Expand}_{\mathsf{P}}(pp_{\mathsf{P}}, pk_{\mathsf{P}}, sk_{\mathsf{P}})$**:** Augment the public key pk_{P} with the description of an efficient hash function $\mathcal{H}$ of range $\mathbb{Z}_N$ and return $(pp^{\mathsf{va}}_{\mathsf{P}}, pk^{\mathsf{va}}_{\mathsf{P}}, sk^{\mathsf{va}}_{\mathsf{P}})$.

$\mathsf{VA.Enc}_{\mathsf{P}}(pk^{\mathsf{va}}_{\mathsf{P}}, m)$**:** Compute $c = \mathsf{Enc}_{\mathsf{P}}(pk_{\mathsf{P}}, m\,; a, s, t)$. Then select random r, b in $\mathbb{Z}_N$ and u, v in $\mathbb{Z}^*_N$, and compute $c' = \mathsf{Enc}_{\mathsf{P}}(pk_{\mathsf{P}}, r; b, u, v)$. Compute $e = \mathcal{H}(pp^{\mathsf{va}}_{\mathsf{P}}, pk^{\mathsf{va}}_{\mathsf{P}}, c, c')$ and then $f = (f_m, f_a, f_s, f_t) = (r + em, b + ea, us^e, vt^e)$. The ciphertext is made of $c^{\mathsf{va}} = (c, e, f)$.

$\mathsf{Valid}_{\mathsf{P}}(pk^{\mathsf{va}}_{\mathsf{P}}, c^{\mathsf{va}})$**:** Parse c^{va} and check whether all elements of the ciphertext are properly encoded.[1] Compute $d' = G^{f_m} H^{f_a} \cdot d^{-e}$, $c'_1 = (N+1)^{f_m} f_s^N \cdot c_1^{-e}$ and $c'_2 = (N+1)^{f_a} f_t^N \cdot c_2^{-e}$, and test whether the following equality holds: $e = \mathcal{H}(pp^{\mathsf{va}}_{\mathsf{P}}, pk^{\mathsf{va}}_{\mathsf{P}}, c, c')$ for $c' = (d', c'_1, c'_2)$. If the verifications fail, output $\perp$ else output 1.

$\mathsf{Strip}_{\mathsf{P}}(pk^{\mathsf{va}}_{\mathsf{P}}, c^{\mathsf{va}})$**:** Following the description of c^{va} provided by pk^{va} parse it as (c, e, f). If that fails output $\perp$, and c otherwise.

The other CCVA algorithms are entirely determined by Definition 5.

By construction this scheme satisfies all the conditions of validity and security of Theorem 4 (in the random oracle model, assuming the hardness of the DCR problem). As a result, for any tallying function ρ, the corresponding minivoting scheme $\mathsf{Enc2Vote}(\mathsf{PPATP}, \rho)$ has a PPAT and guarantees ballot privacy.

This instance of our generic construction is very simple. Unfortunately, it exhibits at least two important limitations for a practical use. First, being based on a Paillier-style cryptosystem, the threshold key generation algorithm either requires the existence of a single trusted party that produces an RSA modulus N and forgets its factorization, or the use of fairly sophisticated multiparty computation protocol to generate this modulus in a distributed way [17]. The first option is often challenging to setup in practice, while the second can often be infeasible as it requires substantial expertise from the key holders.

Furthermore, the computational cost of this scheme can be fairly heavy, especially if one desires to use a homomorphic tallying approach that will usually require to compute a number of modular exponentiations that will be 5 to 10 times higher than the number of candidates [16].

This stands in contrast with encryption schemes based on prime order groups: they enjoy considerably simpler key generation procedures [21], and typically enable much more efficient computation in the underlying groups, which can be of 256-bit instead of 2048-bit order for similar security levels.

As mentioned above, the PPATP scheme is a slight variant of a scheme suggested by Moran and Naor and also used by Demirel et al. [18]. Their version however relies on cut-and-choose techniques to prove the validity of ciphertexts and is therefore considerably less efficient.

6 Efficient CCVAE Schemes

This section describes two much more efficient and usable constructions of CCVAE schemes. These schemes do not follow the generic approach presented in the previous section but combine encryptions and commitments in a more efficient way and make it possible to perform all computation in prime order groups.

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.

6.1 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.

[1] We assume some unique and efficiently verifiable encoding for the elements of our various groups. This is needed in order to avoid some trivial malleability relations.

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

Proof. We first observe that PPATS is a CCVAE scheme. Indeed, the formal aspects of the definition are satisfied, and Valid_S algorithm is sound thanks to the soundness of the σ_{cc} proof and from the binding property of the TC2 scheme. Both these properties hold in the Λ_{sxdh} setting.

To verify that PPATS offers NM-CPA security, we show that a PPATS ciphertext is made of an IND-CPA element c. The NM-CPA security then follows from the fact σ_{cc} is a sigma proof of knowledge of the plaintext and randomness used to compute c.

To show the IND-CPA security of c, we show that it is indistinguishable of a random tuple of (independent) group elements, using the following intermediary distributions. Let m be an element of $\mathbb{Z}_q$.

$\mathcal{D}_3$ is the regular distribution $(h^r h_1^m, g^s, g^r g_1^s)$ produced by encrypting m using random r, s.

$\mathcal{D}_2$ is the distribution $(h^r h_1^m, g^s, g^r g_1^{s'})$ identical to $\mathcal{D}_3$ except that the third element is now computed using a fresh random s'. Any distinguisher between $\mathcal{D}_3$ and $\mathcal{D}_2$ can solve the DDH problem in $\mathbb{G}_1$ with the same success probability.

$\mathcal{D}_1$ is the distribution $(h^r h_1^m, g^s, g^{s'})$, identical to $\mathcal{D}_2$.

$\mathcal{D}_0$ is the distribution $(h^r, g^s, g^{s'})$, which is identical to $\mathcal{D}_1$.

We observe that the 3 elements of $\mathcal{D}_0$ are all random and independent. So, the advantage of any adversary against this encryption scheme has its success probability bounded by the probability of breaking DDH in $\mathbb{G}_1$ by any adversary using the same computational effort (up to the computation of a constant number of modular exponentiations).

The validity augmentation part of PPATS simply add a secure sigma proof. The NM-CPA security of the scheme is obtained by applying Theorem 4. □

Proving vote validity. Considering the minivoting scheme based on PPATS, it is easy to provide additional verifiability mechanism to ensure the validity of a vote. Consider for instance the case of an approval election. In this case, we need to prove that a voter submitted an encryption of $m = 0$ or $m = 1$. This can be done requiring each voter to append to the encryption of her choice a non-interactive 0-1 sigma or-proof σ_{or} on the public commitment part [14]. More precisely, when computing $c^{\mathsf{va}} = (c, \sigma_{cc})$ where $c = (d, c_1, c_2)$ the voter conducts the following extra steps: she computes $w = (w_0, w_1)$ where $w_m = h^b$, $w_{1-m} = h^{t_{1-m}}(d/h_1^{1-m})^{-\nu_{1-m}}$ for random b, ν_{1-m}, t_{1-m} in $\mathbb{Z}_q$, then defines $\sigma_{or} = (\nu_0, \nu_1, t_0, t_1)$ with $\nu_m = \mathcal{H}(pp_{\mathsf{S}}^{\mathsf{va}}, pk_{\mathsf{S}}^{\mathsf{va}}, d, w) - \nu_{1-m}$.

Given the commitment d and the proof $\sigma_{or} = (\nu_0, \nu_1, t_0, t_1)$, anybody can verify that d can only be opened to a vote for 0 or 1: compute $w = (w_0, w_1)$ such that $w_i = h^{t_i}(d/h_1^i)^{-\nu_i}$, for $i = 0, 1$, and check whether $\nu_0 + \nu_1 = \mathcal{H}(pp_{\mathsf{S}}^{\mathsf{va}}, pk_{\mathsf{S}}^{\mathsf{va}}, d, w)$. Furthermore, since the σ_{or} proof is perfect zero knowledge, it can indeed safely appear on the public bulletin board $\boldsymbol{PB}$ without affecting the privacy goals in any way.

Elections with homomorphic tallying from PPATS. We can now use this scheme to build a voting scheme PPATSVote based on Enc2Vote(PPATS, ρ_{S}) but from which we modify the Tally algorithm as follows.

1. *Stripping*: Once the polls are closed, the authorities run $\mathsf{Valid}_{\mathsf{S}}$ and $\mathsf{Strip}_{\mathsf{S}}$ 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 $v = \mathsf{Dec}_{\mathsf{S}}(sk_{\mathsf{S}}, c)$ the result of the election. To prove the correctness of the decryption, they also run $\mathsf{Open}_{\mathsf{S}}$ on c, obtaining an auxiliary value a. Finally the authorities append (v, a) on $\boldsymbol{PB}$.

Theorem 6. *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 Enc2Vote(PPATS, ρ_{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 Enc2Vote(PPATS, ρ_{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. [28] and applied to the Bingo voting scheme and then to several variants of the Helios voting system [27].

While plugging the PPATS scheme into Helios would not have any noticeable impact on the verifiability analysis of Helios proposed by Kremer et al. [26], 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. [27]. 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, 27]. A rigorous cryptographic analysis of verifiability/accountability of a fully-fledged voting system is an open problem (note that all current works on Helios [26, 27] abstracted the cryptographic aspects and, as result, overlooked the recently found attacks on the verifiability of Helios [9]), and is out of our scope.

Variations on PPATS. In the Λ_{sxdh} setting, the group operations in $\mathbb{G}_2$ are typically much more expensive than those in $\mathbb{G}_1$. As a result, it might be more efficient to use a slightly modified version of PPATS for more complex ballot validity proofs (e.g., the vote is an integer in a larger range). Indeed the ciphertext could be extended with an extra element $c_3 = g^m g_2^s$ in $\mathbb{G}_1$ which, together with c_1 would consist of an ElGamal (re-)encryption of m. The validity proofs could then be on c_3, and the decryption algorithm could become faster as well. However, a proof that (d, c_3) is well-formed would also have to be published, which makes that construction more expensive than the PPATS scheme that we consider since the computations of elements in $\mathbb{G}_2$ for the simple 0/1 validity proof have a lower cost.

6.2 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 anymore, 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:*

$\mathsf{VA.Gen}_{\mathsf{C}}(1^n)$**:** 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 $(pp^{\mathsf{va}}_{\mathsf{C}} = pp_{\mathsf{C}}, pk^{\mathsf{va}}_{\mathsf{C}}, sk^{\mathsf{va}}_{\mathsf{C}} = sk_{\mathsf{C}})$.

$\mathsf{VA.Enc}_{\mathsf{C}}(pk^{\mathsf{va}}_{\mathsf{C}}, m\,; r, r_1, r_2)$**:** Compute $c = \mathsf{Enc}_{\mathsf{C}}(pk_{\mathsf{C}}, m; r, r_1, r_2)$, 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}).

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

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

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

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

$\mathsf{Valid}_{\mathsf{C}}(pk^{\mathsf{va}}, c^{\mathsf{va}})$**:** 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 $\mathsf{Strip}_{\mathsf{C}}$ returns c from c^{va} in the obvious way. Applying $\mathsf{Strip}_{\mathsf{C}}$ 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].

As far as efficiency is concerned, note that a voter will never have to compute elements in $\mathbb{G}_T$ and especially pairings when creating a ballot.

Theorem 7. *The* PPATC *scheme is an NM-CPA secure CCVAE scheme in the random oracle model in the* Λ_{sxdh} *setting.*

Proof. This proof follows the scheme of the one of Theorem 5. We first observe that PPATC is a CCVAE scheme. Indeed, the formal aspects of the definition are satisfied, and Valid_S algorithm is sound thanks to the soundness of the σ_{cc} proof and from the binding property of the scheme of Abe et al.[2]. Both these properties hold in the Λ_{sxdh} setting.

To verify that PPATC offers NM-CPA security, we show that a PPATC ciphertext is made of an IND-CPA element c. The NM-CPA security then follows from the fact σ_{cc} is a sigma proof of knowledge of the plaintext and randomness used to compute c.

To show the IND-CPA security of c, we show that it is indistinguishable of a random tuple of (independent) group elements, using the following intermediary distributions. Let m be an element of $\mathbb{G}_1$.

$\mathcal{D}_5$ is the regular distribution $(g^{r_1}, g^{r_2}, g_1^r g_2^{r_2}, h^r h_1^{r_1}, m g_1^{r_1})$ produced by encrypting m using random r, r_1, r_2.

$\mathcal{D}_4$ is the distribution $(g^{r_1}, g^{r_2}, g_1^r g_2^{s_2}, h^r h_1^{r_1}, m g_1^{r_1})$ identical to $\mathcal{D}_5$ except that the third element is now computed using a fresh random s_2. Any distinguisher between $\mathcal{D}_5$ and D_4 can solve the DDH problem in $\mathbb{G}_1$ with the same success probability.

$\mathcal{D}_3$ is the distribution $(g^{r_1}, g^{r_2}, g_2^{s_2}, h^r h_1^{r_1}, m g_1^{r_1})$, which is identical to $\mathcal{D}_4$.

$\mathcal{D}_2$ is the distribution $(g^{r_1}, g^{r_2}, g_2^{s_2}, h^r, m g_1^{r_1})$, which is identical to $\mathcal{D}_3$.

$\mathcal{D}_1$ is the distribution $(g^{r_1}, g^{r_2}, g_2^{s_2}, h^r, m g_1^{s_1})$, obtained by replacing r_1 with a random fresh value s_1 in the 4th element. Again, any distinguisher between $\mathcal{D}_2$ and D_1 can solve the DDH problem in $\mathbb{G}_1$ with the same success probability.

$\mathcal{D}_0$ is the distribution $(g^{r_1}, g^{r_2}, g_2^{s_2}, h^r, g_1^{s_1})$, which is identical to $\mathcal{D}_1$.

We observe that the 5 elements of $\mathcal{D}_0$ are all random and independent. So, the advantage of any adversary against this encryption scheme has its success probability bounded by twice the probability of breaking DDH in $\mathbb{G}_1$ by any adversary using the same computational effort (up to the computation of a constant number of modular exponentiations).

Finally, observe that the NIZK proof σ_{cc} added to obtain the PPATC scheme is perfectly zero knowledge and thus does not break the privacy property. We conclude by applying Theorem 4. □

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 ***PB***. Various perfect (or statistical) ZK proof of shuffles can be used for that purpose [23, 25, 35]: 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 [36, 35] 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 $\boldsymbol{v'}$ as well as the corresponding commitment auxiliary values, and publish these 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.

Verification of the Permutation Commitment Public observers verify the proof of permutation commitment P_π and abort if it fails.

Verification of the proof of shuffle Public observers verify the proof of shuffle $P_{\boldsymbol{d}}$ and abort if it fails.

Verification of the openings Public observers 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.

7 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 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 Appendix A.)

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 bits RSA modulus N. We select $\mathbb{G}_1$ to be taken on $\mathbb{F}_p$ for a 256 bits long prime p and $\mathbb{G}_2$ to be taken on $\mathbb{F}_{p^2}$. The cost of a point addition is evaluated to 16 multiplications in the underlying field, and the

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 bits vote)	3.375	4	0	0	3809280
PPATS (0/1 vote)	0	0	6	6	115200
PPATC (256 bits vote)	0	0	9	4	96000

Table 1. Ciphertext computation workload.

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-bits prime order group in less than 30ms in the Chrome web browser, the computation of a PPATC ciphertext that can encode a 256-bits 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.

A prototype implementation of our PPATS scheme is currently realized in JavaScript. It would be interesting to see how far the efficiency of this scheme can be improved when multiple ciphertexts have to be computed, e.g., by relying on fixed base exponentiation techniques.

References

1. M. Abe, K. Haralambiev, and M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. *IACR Cryptology ePrint Archive*, 2010:133, 2010.
2. M. Abe, K. Haralambiev, and M. Ohkubo. Group to group commitments do not shrink. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 301–317. Springer, 2012.
3. B. Adida, O. de Marneffe, O. Pereira, and J.-J. Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In T. Moran D. Jefferson, J.L. Hall, editor, *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. Usenix, August 2009.
4. Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical everlasting privacy. In *POST*, volume 7796 of *LNCS*, pages 21–40. Springer, 2013.
5. P. S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks*, volume 2576 of *LNCS*, pages 257–267. Springer, 2003.
6. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC'2005*, volume 3897 of *LNCS*, pages 319–331. Springer, 2006.
7. J Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, January 1987.
8. D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi. Adapting helios for provable ballot privacy. In *Computer Security – ESORICS 2011*, LNCS. Springer, 2011.
9. D. Bernhard, O. Pereira, and B. Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT 2012*, number 7658 in LNCS, pages 626–643. Springer, 12 2012.
10. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
11. R. Carback, D. Chaum, J. Clark, J. Conway, A. Essex, P. S. Herrnson, T. Mayberry, S. Popoveniuc, R. L. Rivest, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity ii municipal election at takoma park: The first e2e binding governmental election with ballot privacy. In *USENIX Security Symposium*, pages 291–306. USENIX Association, 2010.
12. D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. L. Vora. Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Security and Privacy*, 6(3):40–46, 2008.
13. J. Cohen (Benaloh) and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of 26th Symposium on Foundations of Computer Science.*, pages 372–382, Portland, OR, 1985. IEEE.

14. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.
15. R. Cramer, F. K. Franklin, B. Schoenmakers, and M. Yung. Multi-autority secret-ballot elections with linear work. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.
16. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.
17. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, volume 1992 of *LNCS*, pages 119–136. Springer, 2001.
18. Denis Demirel, Jeroen van de Graaf, and Roberto Araújo. Improving helios with everlasting privacy towards the public. In A. Halderman and O. Pereira, editors, *EVT/WOTE 2012*. USENIX, 2012.
19. Denise Demirel and Jeroen van de Graaf. A publicly-veri
able mix-net with everlasting privacy towards observers. Cryptology ePrint Archive, Report 2012/420, 2012. `http://eprint.iacr.org/`.
20. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer, 1993.
21. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptology*, 20(1):51–83, 2007.
22. Gurchetan S. Grewal, Mark D. Ryan, Sergiu Bursuc, , and Peter Y. A. Ryan. Caveat coercitor: Coercion-evidence in electronic voting. In *IEEE Security and Privacy Symposium*, 2013.
23. J. Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, 2010.
24. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.
25. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353. USENIX, 2002.
26. S. Kremer, M. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *Computer Security - ESORICS 2010*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.
27. R. Küsters, T. Truderung, and A. Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *IEEE Symposium on Security and Privacy (S&P 2012)*, pages 395–409. IEEE Computer Society, 2012.
28. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: definition and relationship to verifiability. In *CCS '10*, pages 526–535. ACM, 2010.
29. T. Moran and M. Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In *26th International Cryptology Conference (CRYPTO'06)*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006.
30. T. Moran and M. Naor. Split-ballot voting: Everlasting privacy with distributed trust. *ACM Trans. Inf. Syst. Secur.*, 13(2), 2010.
31. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, LNCS, pages 223–238. Springer, 1999.
32. P.Y.A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. The prêt à voter verifiable election system. *IEEE Transactions on Information Forensics and Security*, 4:662–673, 2009.
33. K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *LNCS*, pages 393–403. Springer, 1995.
34. R. G. Saltman. *The history and politics of voting technology.* Palgrave Macmillan, 2006.
35. B. Terelius and D. Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 100–113. Springer, 2010.
36. D. Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *Information Security and Privacy, 14th Australasian Conference, ACISP 2009*, volume 5594 of *LNCS*, pages 407–421. Springer, 2009.

A Efficiency evaluation

Using the methodology provided in Section 7, a point multiplication comes at a cost of approximately 3840 in $\mathbb{G}_1$ and 15360 in $\mathbb{G}_2$. An exponentiation modulo a 2048 bits prime costs 196608 while when it is done modulo a 4096-bit N^2 modulus, the cost rises to 786432. An exponentiation modN^2 is roughly 205 times higher than a point multiplication in $\mathbb{G}_1$. Note that, in some cases, the exponent will have 256 bits instead of 2048 (e.g., when the exponent is a 256 bits vote or the output of a hash function.) We then reduce the cost of exponentiations by a factor 8.

PPATP *for Generic Use.* We consider the cost of one vote with a security equivalent to a 2048 bits RSA. A vote is then made of:

- A Pedersen commitment to the choice of the voter, which takes 1 exponentiation $\bmod P$ for a 1 bit vote, or 1.125 exponentiations $\bmod P$ for a 256 bit vote.
- A Paillier encryption of the vote and the opening costing 2 exponentiations modulo a 4096-bit modulus (note that $(1+N)^x = (1+xN) \bmod N^2$).
- A proof that the ballot is consistent. It takes 2 exp. $\bmod P$ and 2 exp. $\bmod N^2$ and 2 short exponentiations $\bmod N$ (in the proof response).
- If a 0/1 proof must be computed, 2.125 exponentiations $\bmod P$ must be added.

In total we have 5.375 and 4 exponentiations $\bmod P$ or N and $\bmod N^2$, respectively, for a single bit vote + a 0/1 proof. In the case of a 256 bits vote, we obtain 3.375 and 4 exponentiations $\bmod P/N$ and $\bmod N^2$, respectively.

PPATS *with 0/1 proof.* We consider the cost of a 0/1 vote, which is made of:

- A CCE ciphertext that takes 3 (point) multiplications in $\mathbb{G}_1$ and 1 multiplication in $\mathbb{G}_2$ for a 1 bit vote.
- A validity augmentation that takes 3 multiplications in $\mathbb{G}_1$ and 2 in $\mathbb{G}_2$.
- A 0/1 proof of validity of the commitment, which takes 3 multiplications in $\mathbb{G}_2$.

In total, the voter computes 6 multiplications in $\mathbb{G}_1$ and 6 in $\mathbb{G}_2$.

PPATC *with 256 bit votes.* We consider the cost of a 256 bits vote, which is made of:

- A CCE ciphertext that takes 5 multiplications in $\mathbb{G}_1$ and 2 in $\mathbb{G}_2$.
- A validity augmentation that takes 4 multiplications in $\mathbb{G}_1$ and 2 in $\mathbb{G}_2$.

In total, the voter computes 9 multiplications in $\mathbb{G}_1$ and 4 in $\mathbb{G}_2$.