Scaling Privacy Guarantees in Code-Verification Elections

Aggelos Kiayias[1,⋆] and Anthi Orfanou[2]

[1] National and Kapodistrian University of Athens, Athens, Greece
aggelos@kiayias.com
[2] Columbia University, New York, NY
anthi@cs.columbia.edu

Abstract. Preventing the corruption of the voting platform is a major issue for any e-voting scheme. To address this, a number of recent protocols enable voters to validate the operation of their platform by utilizing a platform independent feedback: the voting system reaches out to the voter to convince her that the vote was cast as intended. This poses two major problems: first, the system should not learn the actual vote; second, the voter should be able to validate the system's response without performing a mathematically complex protocol (we call this property "human verifiability"). Current solutions with convincing privacy guarantees suffer from trust scalability problems: either a small coalition of servers can entirely break privacy or the platform has a secret key which prevents the privacy from being breached. In this work we demonstrate how it is possible to provide better trust distribution without platform side secrets by increasing the number of feedback messages back to the voter. The main challenge of our approach is to maintain human verifiability: to solve this we provide new techniques that are based on either simple mathematical calculations or a novel visual cryptography technique that we call *visual sharing of shape descriptions*, which may be of independent interest.

Keywords: Electronic voting, elections integrity, visual cryptography.

1 Introduction

The integrity of the voting platform is a critical feature of electronic voting systems. If an attacker controls the voting platform then it can not only breach voter privacy but also manipulate the election results. For this reason, as e-voting systems increasingly find their way to real-world deployments, the security properties of the voting platform have become a major consideration. This problem is particularly exacerbated in the case of Internet voting where the voter is supposed to use a general purpose system (PC) for ballot casting. In this context

⋆ Supported by project FINER of Greek Secretariat of Research and Technology, ERC project CODAMODA and Marie Curie grant RECUP. Also partly supported by an EAC grant from the VoTeR center - University of Connecticut.

the problem has been generally identified as the *untrusted platform problem.* To solve the problem a general methodology has arisen that enables the human operator of the ballot casting PC to validate its operation (i.e., that it has cast the proper vote) by receiving a suitable feedback from the system. This approach, even if we assume the existence of such feedback channel for free[1], it has to additionally overcome two major challenges: First, the system should be able to provide such feedback without breaching the privacy of the voter and learning its vote. Second, the validation protocol should not be mathematically complex since then this would require the utilization of the PC again to complete it; in other words, the protocol should be "human-verifiable" i.e., easily executed by a human in the verifier side. We first explain how these problems have been addressed in the literature so far and then we present our results.

1.1 Previous Work

An ingenious idea to resolve the untrusted platform problem was proposed by Chaum [4]: in *code voting* the system provides to the voter a code that uniquely corresponds to his proper vote but the actual correspondence is hidden. The system used pre-encrypted ballots and return codes, requiring the voters to enter pseudo-random numbers in order to cast a vote. The scheme guarantees privacy and integrity against malicious computers however codes need to be generated via a multiparty protocol and distributed privately, something that substantially increases the complexity of the system. Later the idea of code voting was applied in voting schemes like those in [23,13].

Subsequent work in code verification protocols cf. [14,11] simplified the ballot casting procedure and made it compatible with standard encrypted ballot casting systems (so that previous tallying mechanisms can be readily applied, as those in [12,1,2,15,20,7]). More specifically, Heiberg et al. propose in [14] a code verification protocol that uses random security codes. These are reconstructed as the vote is transfered from the PC to the messenger through the vote collector, by a proxy oblivious transfer scheme. Gjøsteen in [10,11] uses pseudo-random codes, generated as the composition of three pseudo-random functions, owned by the PC, the vote collector and the messenger respectively. These papers focused on the vote collection and feedback system which is comprised of two servers, a vote collector and a messenger who collaborate to produce the validation code that is transmitted as feedback back to the user. The separation of these two servers is an essential feature and their collaboration breaks privacy in the protocols of [14,11]. To address this serious privacy issue (as well as a few other problems), Lipmaa presented an adaptation of Gjøsteen's protocol in [16], that prevents the coalition of the servers from breaching privacy by relying on a secret stored on the PC. In this case, the vote collector and messenger server coalition is still unable to breach privacy unless somehow they get access to the PC secret-key. While this addresses partially the privacy concern it increases the

[1] For example an SMS to a smartphone has been suggested as an implementation of the feedback mechanism.

key-management requirements of the protocol on the PC side. Given the above, it remains as an open question to provide better privacy guarantees without using a secret key on the PC side.

Part of our techniques to be presented below are related to visual cryptography. Naor and Shamir [19] introduced visual cryptography and proposed a visual secret sharing protocol that shares black and white images. In their system each pixel is mapped to a certain shape and can be subdivided to a number of black and white sub-pixels. The scheme considers a pixel as black if the number of black sub-pixels exceeds a certain threshold and analogously white if the black sub-pixels are below a threshold. The final pixels are revealed visually by overlaying a number of shares. While very interesting, the techniques of visual cryptography have found little applications in real world systems. Chaum exploited visual cryptography for visual receipts in supervised electronic voting (elections through voting booths) [5]. The scheme uses a 2-out-of-2 secret sharing scheme to share the written form of the vote in two complementary sheets that reveal the vote when combined, while none of the sheets leaks information on its own. The voters keep one sheet and verify their ballot by comparing it with a copy posted on the bulletin board. The use of visual cryptography was later found to be non-essential and the original system was simplified in a way that obviates the visual cryptography part, [22].

1.2 Our Results

In this work we tackle the problem of scaling the privacy guarantee in code verification voting systems without requiring any secret-keys on the PC side. Our approach to achieve this is by increasing the number of feedback messages back to the voter in order to enable the distribution of the messenger functionality. The main challenge of this approach is to maintain human verifiability: to solve this we provide new techniques that are based on either a simple mathematical calculation that the voter is supposed to execute or a novel visual cryptography technique that we call *visual sharing of shape descriptions* and is detailed below.

In general we follow the same initial setup as the voting systems of [14,10,16], i.e., the voter interacts with her PC to generate an encrypted vote. This vote however, is transmitted to *a set* of voting servers: in our scalable system we need not distinguish between types of servers as in previous protocols — all of our servers behave in identical fashion. The voting servers provide feedback to the user through an untappable channel (as the single messenger did in the previous protocols cited above). Each feedback by itself carries no information that can be tied to a specific voter choice. Nevertheless, the voter is able to validate her vote by appropriately synthesizing the feedback she receives from the servers. We consider the cases where the feedback may be the vote itself, a visual representation of the vote (explained below) or a voter dependent security code. In the latter case it is also required to have another out-of-band channel for the distribution of the security codes (as in [14,11,4]). The first two cases though, highlight a unique feature of our methodology: since there is a set of voting servers that each one of them is incapable of extracting something useful from the feedback they return to the voter, it is actually possible for the synthesized feedback to be the actual vote that was casted.

We give two constructions that address the problem of human verifiability (via the synthesis of the voting server feedback messages). In the first case we assume that the voter is capable of executing addition of decimal numbers, i.e., the voter should be capable of verifying that, e.g., the two-digit decimal numbers 32 and 92 sum up to a number that ends in 44. While for some people this may be an easy task, it can be argued by some that it is a tall order for the general population who may not be accustomed to perform mathematical calculations. For this reason we introduce an entirely different technique that is related to (but distinct from) visual cryptography and may be of independent interest.

A visual sharing of shape descriptions is a way to perform secret-sharing of a set of *shape descriptions.* For example a shape description can be the following: "a full circle." In this setting the voter may be given two or more images and we assume that she is capable of deciding (in her mind) whether the overlay of the images matches the shape description she is given. In the case of a full circle, for example, a question that the voter is assumed to be able to answer is the following: does the overlay of ◐ and ◑ amount to a full circle? In our second vote verification protocol the voter validates the PC operation by answering queries such as this one.

We present our protocols for the case that the voting server feedback synthesizes back to the actual vote, however, as mentioned, our protocols can be easily adapted to the code verification setting (as in [14,11]). In this setting a code generation phase takes place before the elections and the codes are sent to the voter through an out-of-band communication channel (called the pre-channel that for instance is paper mail sent ahead of the elections). Then, the voting servers will obtain a *share* of the code that corresponds to the submitted vote and will forward it, through another out-of-band channel (the post-channel), to the voter as feedback that will be synthesized as above using our techniques. An attacker may view the contents of at most one of these channels, in order to guarantee privacy.

2 Cryptographic Preliminaries and Tools

Public Key Cryptosystem. A public key cryptosystem is a triple of algorithms $\langle Gen, Enc, Dec \rangle$. The randomized Gen algorithm on input the security parameter 1^k outputs a secret/public key pair $(pk, sk) \leftarrow Gen(1^k)$. The Enc_{pk} randomized algorithm on input pk, a message m and randomness r outputs a ciphertext $c = Enc_{pk}(m, r)$. The deterministic Dec_{sk} algorithm on input sk and a ciphertext $c \in C$ outputs a message $m' = Dec_{sk}(c)$. For a correct encryption scheme it holds that if $(pk, sk) \leftarrow Gen(1^k)$ then $Dec_{sk}(Enc_{pk}(m, r)) = m$.

The ElGamal cryptosystem works over a finite cyclic group G_q of prime order q, generated by $\langle g \rangle$. Gen selects a secret key $sk \leftarrow \mathbb{Z}_q$ and sets $pk = g^{sk}$. A message $m \in G$ is encrypted as $\langle c_1, c_2 \rangle = Enc_{pk}(m, r) = \langle m \cdot pk^r, g^r \rangle$, with randomness $r \in \mathbb{Z}_q$. On input $\langle c_1, c_2 \rangle$ the decryption algorithm outputs

$m' = Dec_{sk}(\langle c_1, c_2\rangle) = c_1/c_2^{sk}$. The cryptosystem is multiplicatively homomorphic as for all $m_1, m_2, r_1, r_2 \in \mathbb{Z}_q$ it holds that $Enc_{pk}(m_1, r_1) \cdot Enc_{pk}(m_2, r_2) = Enc_{pk}(m_1 \cdot m_2, r_1 + r_2)$. An additively homomorphic variant is derived if we encrypt g^m instead of m. The ElGamal cryptosystem is IND-CPA secure under the decisional Diffie-Hellman assumption.

Commitments. A commitment scheme is a triple of algorithms $\langle Gen, Com, Open\rangle$. The randomized Gen algorithm on input 1^k outputs a public key h. The randomized Com_h algorithm on input h, a message m and randomness r and outputs a commitment $c = Com_h(m, r)$. The $Open$ algorithm on input a commitment c and the de-commiting values m, r verifies that $c = Com_h(m, r)$. A commitment scheme satisfies the statistically hiding property if the distributions of commitments for two different messages are indistinguishable. It satisfies the computationally binding property if any polynomial-time adversary cannot open a commitment to two different values with non-negligible probability. The Pedersen commitment scheme [21] works over a finite cyclic group G_q of prime order q generated by $\langle g\rangle$. The message and randomness space is $\mathbb{Z}_q$ and the cipherspace G_q. $Gen(1^k)$ outputs a key $h = g^\alpha$ for $\alpha \leftarrow \mathbb{Z}_q$ and algorithm Com_h, on input $m, r \in \mathbb{Z}_q$ outputs $c = g^m h^r$. The scheme is statistically hiding and computationally binding under the decisional Diffie-Hellman assumption.

Signatures. A digital signature scheme is a triple of algorithms $\langle Gen, Sign, Ver\rangle$. The randomized Gen algorithm on input 1^k outputs a verification/signing key pair $(vk, sk) \leftarrow Gen(1^k)$. The randomized $Sign_{sk}$ algorithm on input the signing key sk, a message m and randomness r outputs a signature $\sigma = Sign_{sk}(m, r)$ and the Ver_{vk} algorithm, on input the verification key vk, a message m and a signature σ accepts the signature as valid or rejects it. Security for signatures is defined as existentially unforgeability against chosen message attack (EUF-CMA), stating that no polynomial forger can produce a valid signature for a message that he has not seen before, assuming black box access to a signing oracle. For our purposes we rely on any EUF-CMA signature scheme and we assume the existence of a public key infrastructure that can be used to digitally sign messages. All participants of the protocol, i.e. the voter's PCs and the online voting servers, are assumed to support these operations.

Proofs of Knowledge. A proof of knowledge is a communication protocol between two entities, a Prover and a Verifier. The prover possesses a valid witness w for a publicly known predicate R, such that $R(w) = 1$ and wants to convince the verifier without revealing the witness. A special case of proofs of knowledge are the 3-message Σ-protocols whose communication transcript is of the form $\langle com, chl, ans\rangle$. The prover makes the first step to send a commitment com to the verifier. The response of the verifier is a randomly chosen challenge chl. The prover terminates the protocol by sending an answer message ans and the verifier checks the validity of some conditions. Any Σ-protocol can be made non interactive in the random oracle model by using the Fiat-Shamir heuristic [8]. The techniques of [6] allow us to produce conjunctions and disjunctions of Σ-protocols that satisfy special soundness and honest-verifier zero-knowledge.

The well known Schnorr protocol for proving knowledge of a discrete logarithm forms the basis of all necessary proofs of knowledge we discuss. Bit commitment proofs of the form $pk(r \mid h^r = C \ \vee \ h^r = (C/g))$, for Pederesen bit commitments C on a public key h, are an immediate consequence of the disjunctions of Schnorr protocols, known as Schnorr "OR" proofs. The proof $pk(\alpha, r_1, r_2 \mid C_1 = g^\alpha h_1^{r_1} \wedge C_2 = g^\alpha h_2^{r_2})$ that two Pedersen commitments C_1, C_2 over the public keys h_1, h_2 hide the same value α, is also derived from Schnorr's protocol. By employing the techniques of [6] the previous proof can be generalized to "OR" proofs for the statement $pk(\alpha, \beta, r_1, r_2 \mid C_1 = g^\alpha h_1^{r_1} \ \wedge \ C_2 = g^\beta h_2^{r_2} \ \wedge \ (\alpha = \beta \ \vee \ \alpha = \beta + u \ \vee \ \ldots \ \vee \ \alpha = \beta + \lambda u))$, stating that the hidden values α, β of the two commitments satisfy the relation $\alpha = \beta + iu$, for some $i \in \{0, \ldots, \lambda\}$ and a publicly known value u.

Finally range proofs are proofs of knowledge showing that a committed value x in a commitment C lies in a specific range of values, such as $[0, m-1]$, for $m \geq 2$. For Pedersen commitments such a proof will be denoted as $pk(\alpha, r \mid C = g^\alpha h^r \ \wedge \ x \in [0, m-1] \ )$. For the purposes of our protocol we employ the range proof from [17]. Alternatively one could use any efficient range proof in exponents, like the generalization of [17] presented in [3]. The proof modifies the classic bit-length range proof of [18] to arbitrary ranges. The proof of [17] writes number $\alpha \in [0, m-1]$ in the form $\alpha = \sum_{j=0}^{\lfloor \log_2(m-1) \rfloor} \mu_j H_j$, where $H_j = \lfloor (m - 1 + 2^j)/2^{j+1} \rfloor$ and $\mu_j \in \{0, 1\}$. Then it commits to all values μ_j and uses bit commitment proofs to show that $\mu_j \in \{0, 1\}$, requiring $k = \lfloor \log_2(m-1) \rfloor + 1$ single bit proofs. For small values of m the proof remains efficient for our purpose. Both the prover and the verifier precompute the coefficients H_j and the verifier can confirm that the committed values μ_j represent α by checking that $g^\alpha = \prod_{j=0}^{k-1} (g^{\mu_j})^{H_j}$.

The Communication Channels. We require the existence of secure communication channels for vote verification. We use the term "untappable channel" to refer to a private channel that prevents an adversary from intercepting sent messages, keeping the information sent through this channel perfectly secret to all other parties. We assume the existence of an one-way untappable channel from the voting servers to the voter to transfer the receipts. This channel can be viewed either as a unique untappable channel used by all servers, or alternatively as a set of communication channels (one from each server to the voter) requiring that one of them should be untappable. Two channels are referred as "a pair of out-of-band communication channels" when they are both secure, authenticated and independent of the PC. In our case we will need a channel from the elections authorities to the voters for receipt distribution and a channel from the voting servers to the voters for verification. For both "out-of-band" channels we prohibit the attacker from modifying their contents. However we may allow the attacker to read the contents of at most one of these channels. We also require the existence of a broadcast channel between the PC and the voting servers, where the PC posts public information required by them.

3 The Vote Verification Protocol

The Security Model. We define the notion of security of our scheme in terms of privacy and integrity. Throughout our discussion we refer to malicious entities. In our setting a malicious PC wants to violate integrity by modifying the vote. We recall that privacy is not relevant against such an attacker, since the PC knows the vote. A set of malicious (honest but curious) voting servers want to violate privacy by learning the vote, as by their construction they cannot alter encrypted submitted ballots. We ask that the following requirements are met:

Cast as intended: We consider the following game between two entities, an adversary A and an honest challenger C: We give A access to the public keys PK and the voter identities ID, and in the code verification setting to the verification codes CS possessed by the voters. A picks a voter V from the ID set, corrupts her PC and lets V cast a ballot for candidate x. Then C runs the whole voting protocol and outputs the encryption of a vote E, the secret receipt R and the public auxiliary information Pub of the protocol. Let Q be a predicate that on input the receipt R and the public information Pub outputs 1 iff R is consistent with Pub. In the code verification setting the codes CS are also part of Q's input, whose output is 1 iff all its input arguments are consistent. A wins the game if $Q(R, Pub, (CS)) = 1$ and $Dec_{sk}(E) \neq x$. A voting protocol with receipts satisfies the "cast as intended" property if it holds that $Pr[A(PK, Pub, ID, (CS))$ wins$] \leq \epsilon(k)$, where $\epsilon(k)$ is a negligible function in the security parameter k.

(t,n)-*Vote secrecy:* We consider the following game between an adversary A and an honest challenger C: We give A have access to the public keys set PK, to the voter identities ID and, in the code verification setting, to the codes CS possessed by the voters. A picks and corrupts $t < n$ out of n servers. A picks a voter identity from the ID set and two candidates x_0, x_1 of his choice and gives them to the challenger. Then C runs the vote casting by picking at random a bit $b \leftarrow \{0,1\}$ and encrypting message x_b as $E = Enc_{pk}(x_b)$. Then C runs the voting protocol and outputs the encrypted vote E, the secret receipt R and all other auxiliary public information and secret information Pub, Sec, sending E, Pub and the appropriate share of Sec_i to server S_i. A in possession of Pub and the private values $\{Sec_i\}_{i \in [j_1,\ldots,j_t]}$ of t compromised servers, outputs a bit b^*. A wins the game if $b^* = b$. A voting protocol satisfies "(t,n)-vote secrecy" if it holds that $Pr[A(PK, Pub, ID, (CS), E, \{Sec_i\}_{i \in [j_1,\ldots,j_t]})$ wins$] \leq 1/2 + \epsilon(k)$, where $\epsilon(k)$ is a negligible function.

As we mentioned before, our solution focuses only on the vote submission phase, like previously suggested protocols [14,16]. The final stage of tallying is considered a separate procedure and correct tallying can be guaranteed by employing a suitable protocol. To address coercion one may allow revoting. However, similarly to previous approaches [14,11,16], this clashes with the cast as intended property since there is no means to guarantee that the servers will send to the tallier the most recent vote submitted by a voter. We note that in case of a wrong receipt we accuse the PC of being malicious (since it is assumed to be the most vulnerable component). Note that if the voting servers' goal is to break privacy, sending wrong receipts will not be useful to them. Still if a server wants to disrupt the election it can create confusion by not issuing receipts, however a voter that verifies her vote will notice that an error has occurred.

3.1 Instantiation of the Vote Verification Protocol

We are now ready to describe the vote verification protocol. Let $n \geq 2$ be the desired number of voting servers and M be the set of m candidates that participate in the elections, represented as globally known elements in $\mathbb{Z}_m$. Moreover let G_q be a subgroup of $\mathbb{Z}_p$ of prime order q over which we implement ElGamal encryption and Pedersen commitments. Our message space is $\mathbb{Z}_u$, where u is an additional system parameter. Specifically u is chosen so as to facilitate the vote reconstruction and verification by the voter. We set $u = \min_\lambda 10^\lambda$ such that $m \leq 10^\lambda < q$. As we consider small scale elections with at most a few hundred options in total, typical values for u will be 100 or 1000. By this trick we avoid the modular additions that would be otherwise required by the vote verification step of the voter, which is simplified to addition of λ-digit decimal numbers. By introducing n voting servers ($2 \leq n < q$), the voter needs to add the corresponding n numbers.

Let us consider the ElGamal key pairs (pk_t, sk_t) of the tallier and the commitment scheme (g, h) that are generated in a key generation phase prior to the elections. During vote submission a voter casts her ballot through her PC voting for candidate $x \in \mathbb{Z}_m$. Then the PC splits the vote by picking $n-1$ random shares $x_1, \ldots, x_{n-1} \in \mathbb{Z}_u$ and adjusting x_n such that $x = \sum_{i=1}^{n} x_i \bmod u$. The PC computes the commitments $(C_1, \ldots, C_n) = (g^{x_1}h^{r_1}, \ldots, g^{x_n}h^{r_n})$ to the shares and sends them, through the broadcast channel, to the voting servers $S_1, \ldots, S_n$ along with the encrypted vote $E_t = (E_x, E_t) = (g^x pk_t^r, g^r)$. The PC needs to prove in zero knowledge that the shares and the vote satisfy the relation $x = \sum_{i=1}^{n} x_i \bmod u$, and opens the commitment C_i to server S_i who verifies its validity. In addition the PC needs to prove that the encrypted candidate corresponds to a valid value in the range $[0, m-1]$. By this we prevent a malicious PC from submitting forged ballots of the form $y = x + ku$ that would yield a correct receipt modulo u.

The PC prepares a non-interactive witness indistinguishable proof of knowledge of the above statements denoted as $\pi = PK(x, r, \{x_i, r_i\}_{i=1}^{n} \mid E_x = g^x pk_t^r \wedge x \in [0, m-1] \wedge \{C_i = g^{x_i}h^{r_i}\}_{i=1}^{n} \wedge x = \sum_{i=1}^{n} x_i \bmod u)$, using standard variations of the Schnorr proof and adapting the techniques of [6]. We note that the proof requires that $n \cdot u < q$ to work properly. From the results of [6] it follows that the proof satisfies correctness, special soundness and honest verifier zero knowledge, however we provide a security proof in appendix D. In our instantiation we will use the non-interactive version of the proof by using the Fiat-Shamir heuristic [8]. Each online server S_i verifies the proof π, decrypts and obtains the share x_i and verifies compatibility with commitment C_i. Upon successful verification of all these steps, the server sends the value x_i through the untappable channel to the voter who verifies the vote by performing a regular addition with possible carry drop beyond the most significant digit ($x = \sum_{i=1}^{n} x_i \bmod 10^\lambda$). In this protocol we allow re-voting as a measure against vote coercion.

The Proof of Knowledge π: Public Input: $\langle p, q, g, u\rangle$ the system parameters, h, pk_t the commitment key and the tallier's public key, m the number of candidates and $k = \lfloor \log_2(m-1)\rfloor + 1$, $E_t = (E_x, E_r) = (g^x pk_t^r, g^r)$, $\{C_i = g^{x_i} h^{r_i}\}_{i=1}^n$. Prover's Input: $x, r, \{x_i, r_i\}_{i=1}^n$.

1. The Prover:
   (a) *Range proof:* For $j = 0, \dots, k-1$ computes $\mu_j \in \{0, 1\}$ s.t. $x = \sum_{j=0}^{k-1} \mu_j H_j$ where $H_j = \lfloor ((m-1) + 2^j)/2^{j+1} \rfloor$
   (b) *Range proof:* For $j = 0, \dots, k-1$:
      i. Picks $z_j \leftarrow \mathbb{Z}_q$ s.t. $\sum_{j=0}^{k-1} z_j H_j = r$.
      ii. Commits to μ_j as $\mathcal{E}_j = g^{\mu_j} pk_t^{z_j}$.
      iii. If $\mu_j = 0$ it picks $w_j, c_{2j}, \rho_{2j} \leftarrow \mathbb{Z}_q$ and sets $y_{1j} = pk_t^{w_j}$, $y_{2j} = pk_t^{\rho_{2j}} (\mathcal{E}_j/g)^{-c_{2j}}$.
      iv. if $\mu_j = 1$ it picks $w_j, c_{1j}, \rho_{1j} \leftarrow \mathbb{Z}_q$ and sets $y_{1j} = pk_t^{\rho_{1j}} (\mathcal{E}_j)^{-c_{1j}}$, $y_{2j} = pk_t^{w_j}$.
   (c) *Valid shares:* If $\sum_{i=1}^n x_i = x + (i-1)u \bmod q$, with $i \in \{1, 2, \dots, n\}$, it picks w, ρ_a, ρ_b, $\{c_j, s_j, \rho'_{1j}, \rho'_{2j}\}_{j \neq i} \leftarrow \mathbb{Z}_q$ and sets $a_i = g^w pk_t^{\rho_a}$, $b_i = g^w h^{\rho_b}$, $\{a_j = (E_x)^{-c_j} g^{s_j} pk_t^{\rho'_{1j}}, b_j = (\prod_{l=1}^n C_l / g^{(j-1)u})^{-c_j} g^{s_j} h^{\rho'_{2j}}\}_{i \neq j}$.
   (d) It sends $(\{a_i, b_i\}_{i=1}^n, \{\mathcal{E}_j, y_{1j}, y_{2j}\}_{j=0}^{k-1})$ to the Verifier.
2. The Verifier picks $c \leftarrow \mathbb{Z}_q$ and sends it to the Prover.
3. The Prover:
   (a) *Range proof:* For $j = 0, \dots, k-1$:
      i. If $\mu_j = 0$ it sets $c_{1j} = c - c_{2j}$, $\rho_{1j} = w_j + c_{1j} z_j$.
      ii. if $\mu_j = 1$ it sets $c_{2j} = c - c_{1j}$, $\rho_{2j} = w_j + c_{2j} z_j$.
   (b) *Valid shares:* If $\sum_{i=1}^n x_i = x + (i-1)u \bmod q$, with $i \in \{1, 2, \dots, n\}$, it sets $c_i = c - \sum_{i \neq j} c_j$, $s_i = w + x c_i$, $\rho'_{1i} = \rho_a + r c_i$, $\rho'_{22} = \rho_b + (\sum_{l=1}^n r_l) c_i$.
   (c) It sends $(\{c_i, s_i, \rho'_{1i}, \rho'_{2i}\}_{i=1}^n, \{c_{1j}, c_{2j}, \rho_{1j}, \rho_{2j}\}_{j=0}^{k-1})$ to the Verifier.
4. The Verifier accepts if all the following tests succeed, otherwise it rejects:
   (a) *Range proof:* For $j = 0, \dots, k-1$: $c = c_{1j} + c_{2j}$ and $pk_t^{\rho_{1j}} = y_{1j} (\mathcal{E}_j)^{c_{1j}}$ and $pk_t^{\rho_{2j}} = y_{2j} (\mathcal{E}_j/g)^{c_{2j}}$.
   (b) *Range proof:* $E_x = \prod_{j=0}^{k-1} \mathcal{E}_j{}^{H_j}$.
   (c) *Valid shares:* $c = \sum_{i=1}^n c_i$ and for $i = 1, \dots, n : g^{s_i} pk_t^{\rho'_{1i}} = a_i (E_x)^{c_i}$ and $g^{s_i} h^{\rho'_{2i}} = b_i (\prod_{l=1}^n C_l / g^{(i-1)u})^{c_i}$.

The Vote Verification Protocol: Let M be the set of m candidates, n the number of servers, $(pk_{S_i}, sk_{S_i}), (pk_t, sk_t)$ be the public/secret key pairs of server S_i $(i = 1, \dots, n)$ and the tallier respectively, (sk_V, vk_V) and (sg_{S_i}, vk_{S_i}) be the signing/verification key pairs of voter V and server S_i, h the commitment public key and $\langle p, q, g, u\rangle$ the system parameters.

– The voter V:
  1. Submits a vote for candidate $x \in \mathbb{Z}_m$.
  2. Waits for shares $x_1, \dots, x_n$ from the servers and checks that $x = x_1 + \dots + x_n \bmod u$.
– The PC $(sk_V, pk_t, \{pk_{S_i}\}_{i=1}^n)$ on input x by the voter:

1. Picks $x_1, \ldots, x_{n-1} \leftarrow \mathbb{Z}_u$ and sets $x_n = x - \sum_{j=1}^{n-1} x_j \bmod u$.
2. Picks $r \leftarrow \mathbb{Z}_q$ and encrypts x as $E_t = (E_x, E_r) = Enc_{pk_t}(g^x, r)$.
3. For all$i = 1, \ldots, n$ picks $\rho_i, r_i, r'_i \leftarrow \mathbb{Z}_q$ and encrypts x_i as $e_i = Enc_{pk_{S_i}}$ (x_i, ρ_i) and commits to it as $C_i = g^{x_i}h^{r_i}$. Then it encrypts the randomness r_i as $R_i = Enc_{pk_{S_i}}(r_i, r'_i)$.
4. Prepares the non interactive proof $\pi = PK(x, r, \{x_i, r_i\}_{i=1}^n \mid E_x = g^x pk_t^r \ \wedge \ \{C_i = g^{x_i}h^{r_i}\}_{i=1}^n \ \wedge \ x = \sum_{i=1}^n x_i \bmod u \ \wedge \ x \in [0, m-1])$.
5. Signs the vote $\sigma = Sing_{sk_V}(E_t, \pi)$.
6. For all $i = 1, \ldots n$ sends to Server S_i $D_i = (e_i, R_i)$ and sends $B = (E_t, \{C_j\}_{j=1}^n, \pi, \sigma, V)$ through the broadcast channel to all servers.

– The Server S_i $(sg_{S_i}, sk_{S_i}, vk_V)$ on input D_i, B performs the following tests. If any step fails it declares a forgery and stops:
  1. Decrypts the de-committing values of D_i to obtain r_i, x_i, verifies the valid opening of $C_i = g^{x_i}h^{r_i}$ and that $x_i \in \mathbb{Z}_u$.
  2. Verifies the proof π and the signature σ.
  3. Signs the vote E_t, $\sigma' = Sign_{sg_{S_i}}(E_t)$, stores it and sends x_i to the voter V through the untappable channel.

– The Tallier: When the election is over the tallier gets the signed votes from a sever, verifies the server's signatures and runs a decryption protocol.

3.2 Security and Performance

We now discuss the security offered by the vote verification protocol. We guarantee that the protocol meets our security requirements in the corruption scenario where the voter's PC is corrupted or a subset of $t \leq n-1$ out of n voting servers are honest-but-curious, i.e. they follow the protocol but share their information with the attacker. We state that if a voter successfully verifies her vote and does not revote, then we guarantee that the vote was cast as intended and remains secret from the voting servers.

Cast as intended: In order for a corrupted PC to succeed in submitting a message $x' \in \mathbb{Z}_q$ instead of a valid candidate $x \in [0, m-1]$ selected by the voter, it must be the case that the receipts are equal, that is $x = x' \bmod u$. Thus x' must be of the form $x' = x + ku \bmod q$, for a $k \in \mathbb{Z}$. Since we assume the execution of the voting protocol by the honest challenger all voting servers check that they receive compatible values in the correct range. Then the range proof guarantees that $x' \in \mathbb{Z}_m \subset \mathbb{Z}_u$, and thus $k = 0$ giving us $x' = x \bmod q$.

(t, n)-*Vote secrecy:* Without loss of generality assume that the adversary controls the first $t < n$ servers and let $x_0, x_1 \in [0, m-1]$ be the candidates chosen by the adversary and given to the voting system. The challenger choses b from $\{0, 1\}$ and generates the n shares $x_{b1}, \ldots, x_{bn} \in \mathbb{Z}_u$, forming the receipt $x_b = \sum_{i=1}^n x_{bi} \bmod u$. The adversary obtains the the private shares $x_{b1}, \ldots, x_{bt}$ of the receipt and the publicly announced vales $C_i = g^{x_{bi}}h^{r_i}$ for all $i = 1, \ldots, n$ and $E = (g^{x_b}pk^r, g^r)$. Since we use a (n, n)-secret sharing scheme the adversary needs all n shares to reconstruct the receipt, while with fewer he obtains a random value $\sum_{i=1}^t x_{bi}$ mod u in $\mathbb{Z}_u$. From the statistically hiding property of Pedersen commitments and ElGamal encryption it is guaranteed that he cannot distinguish between x_0, x_1 from E or from the commitments C_i, for $i = 1, \dots, n$.

We cannot guarantee security against coalitions of malicious computers and malicious voting servers, as in this scenario the malicious voting servers trivially learn the vote from the PC. The PC submits a fake ballot undetected as long as it collaborates with one malicious server who deviates from the protocol and sends a modified value to the voter so that she reconstructs a correct receipt.

Complexity. We calculate the complexity of the n-server protocol, $(2 \leq n < q)$ for m candidates, counting the number of the *online* exponentiations, signings and signature verifications. Values that can be pre-computed like H_j for the range proofs, the votes g^x and the values g^{iu} that appear in the proofs are not counted. The PC performs $6n + 2$ exponentiations for vote encryption and commitments to the shares, $4k$ for the range proof and $5(n - 1) + 3$ for the valid share proof, a total of $4(\lfloor \log_2(m - 1) \rfloor + 1) + 11n$ exponentiations. In addition it performs a single signing. Each server S_i performs 4 exponentiations for decryptions and commitment verifications, $5k$ for verifying the range and $5n$ for verifying the valid shares, a total of $5(\lfloor \log_2(m-1) \rfloor + 1) + 5n + 4$ exponentiations. Additionally the sever performs one signing and one signature verification.

3.3 Instantiation of a Code Verification Protocol

With a simple adaptation our protocol can be transformed to use voter dependent security codes as receipts, relaxing the untappable channel requirement. Following the approach of [14,10,16], we assume a code generation phase before the elections and we use a pair of out-of-band channels, a pre-channel for code delivery to the voters and a post-channel for sending the receipts to the voters.

We sightly change the protocol by creating the security codes through an one-time pad scheme. For each voter V we pick a random value $b_V \in \mathbb{Z}_u$ and set $Code_V[x] = x + b_V \bmod u$, for all candidates x. We also pick $n - 1$ random values $b_{V_1}, \dots, b_{V_{n-1}} \in \mathbb{Z}_m$ and set appropriately b_{V_n} such that $\sum_{i=1}^{n} b_{V_i} = b_V$. We send through the pre-channel the pairs $(x, code_V[x])$ to the voter V. Also the pairs (V, b_{V_i}) need to be given to each server S_i. The voting protocol remains the same, except for the servers' last step, where they send $\alpha_i = x_i + b_{V_i} \bmod u$ through the post-channel. Finally the voter reconstructs the security code as $c = \sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} x_i + b_{V_i} = x + b_V$ and compares it to the value $Code_V[x]$.

We do not allow revoting as the security of the one-time pad scheme collapses otherwise. The protocol has the same security guarantees with the vote verification protocol and additionally an observer of the pre-channel gets no information about the voting process and an observer of the post-channel gets no information about the vote x from the code $c = x + b_V \bmod u$, due to the perfect secrecy of one-time pad. To allow revoting we should use an untappable post-channel. Although in this case we do not relax the untappable channel assumption, voter-dependent receipts can still be useful as a means against coercion.

4 A Visual Vote Verification Protocol

In this section we introduce visual vote verification. This method enables the generation of visual receipts that even though individually leak no information about the submitted vote, overlaying them enables the verification of it. Even though the notion is general, here we will consider the case of two voting servers and an untappable channel to forward the receipts, allowing revoting.

First we introduce a formal cryptographic primitive that we call n-Visual Sharing of Shape Descriptions (n-VSSD). The main idea is that each voter choice will correspond in a unique way to a certain shape. Shapes can be split in n different parts, without revealing information about the initial image; still a person can easily verify that the parts overlay back to the shape. Using a VSSD we show how visual vote verification can be facilitated. Due to lack of space the proofs of this section can be found in appendix B.

The n-VSSD. Let M be the set of $m \geq 2$ distinct messages which we want to represent visually and share in $n \geq 2$ different parts. Let D_x be the set of visual descriptions for message $x \in M$ (we note that we allow each message to have more than one visual representation, i.e., $|D_x| \geq 1$). Also let Λ be a commutative semigroup equipped with an operation $\vee$ that will be called the visual alphabet. The splitting function is a probabilistic function $P : M \to \Lambda^n$, that given a message $m \in M$ outputs a valid "splitting" of it, consisting of n shares in Λ. Then we ask for the following properties:

– Solvability: $\forall x \in M\ \forall \langle v_1, \ldots, v_n \rangle \in P(x)$ it holds that $\vee_{i=1}^{n} v_i \in D_x$.
– (t,n)-Resilience: Let w be an n-tuple of the form $\langle (A \cup \{\#\}), \ldots, (A \cup \{\#\}) \rangle$, where the symbol $\#$ represents a share of unknown value in Λ and A represents known shares in Λ, such that w has $t < n$ known shares A (i.e., different than $\#$). Then there is a constant $0 < c < 1$ such that for all w of the previous form it holds that $Prob_{v \leftarrow P(x)}[w \in v] = c$.

The solvability property corresponds to correctness, requiring the correct reconstruction of a visual description of the initial message from all its n shares. The (t,n)-resilience property corresponds to threshold security, stating that any observer in possession of t or less shares of an image cannot distinguish between the initial message they may belong to, as the ordered t shares can be part of any message with equal probability.

4.1 A 2-VSSD Instantiation

Let us first consider two shape descriptions: *full circle* and *half circle*. These two shape descriptions can be visually represented by a completely black circle and by a circle that is half white in its left or right part. We can correspond these two shapes to a set of messages $M = \{0,1\}$ say, such that, 0 corresponds to full circle and 1 corresponds to half circle. Next we define the visual alphabet Λ to contain two half circles as defined above: half white in the left and half white in the right part. Observe now that $P(0)$ may contain any pair of the two complementary elements from Λ, while $P(1)$ may contain any pair of elements from Λ where the same half circle appears twice.

As bitstrings, we can denote the full circle by 11, while the half-circle by 10 or 01, thus having $D_0 = \{11\}$ and $D_1 = \{01, 10\}$. Our alphabet is $\Lambda = \{01, 10\}$ and the operation $\vee$ is the bitwise OR operation in the elements of Λ applied coordinate-wise. In this case the message corresponding to a shape $bb' \in \cup_{x \in M} D_x$ can be simply recovered as $x = b \oplus b'$ (where $\oplus$ stands for the x-or operation). The splitting function is then defined as follows: $P(0)$ is uniformly chosen from $\{(10, 01), (01, 10)\}$ while $P(1)$ is uniformly chosen from $\{(01, 01), (10, 10)\}$.

Proposition 1. *Let* $M = \{0, 1\}$*. The* $(M, \{D_x\}_{x \in M}, \Lambda, P)$ *scheme defined above is a* 2*-VSSD that satisfies* $(1, 2)$*-resilience.*

A Scheme for Arbitrary M. Let the set M be an arbitrary set of size $m \geq 2$ (corresponding, say, to m different election candidates). Without loss of generality consider $M = \mathbb{Z}_m$. Let $\langle M^*, \{D^*_x\}_{x \in M^*}, \Lambda^*, P^* \rangle$ be the 2-VSSD defined above and $k = \lfloor \log_2(m - 1) \rfloor + 1$. Let $\Lambda = \{10, 01\}^k$ and let $\vee$ be the bitwise OR operation. We next need to determine the visual description set D_x and the splitting function $P(x)$ for each $x \in M$. Let $(b_{k-1} \dots b_0) \in \{0, 1\}^k$ be the binary encoding for a message $x \in M$. We define the set D_x by processing each bit $b_j \in M^* = \{0, 1\}$ of x separately and independently from the others. Specifically a bitstring $(d_{k-1} \dots d_0)$ is in D_x iff for all $j = 0, \dots, k - 1$ $d_j \in D^*_{b_j}$. Similarly the function $P(x)$ applies in each bit of x the splitting function P^* such that the tuple $(a_{k-1} \dots a_0, a'_{k-1} \dots a'_0)$ is in $P(x)$ iff for all $j = 0, \dots k - 1$ $(a_j, a'_j) \in P^*(b_j)$. We provide a detailed example in appendix A.

message bit	shape description	visual	split 1	split 2
0	full circle	●	(10,01): ◐ ◑	(01,10): ◑ ◐
1	half circle	◑ or ◐	(01,01): ◑ ◑	(10,10): ◐ ◐

Fig. 1. The 2-VSSD for $M^* = \{0, 1\}$ with visual alphabet Λ^*

Proposition 2. *For any* $m \in \mathbb{N}$*, the scheme* $(M = \mathbb{Z}_m, \{D_x\}_{x \in M}, \Lambda, P)$ *defined above is a* 2*-VSSD that satisfies* $(1, 2)$*-resilience.*

4.2 Instantiation of the Visual Vote Verification Protocol

Let us proceed to the visual vote verification protocol that uses the above 2-VSSD. As usual the voter votes for candidate x through her PC, which encrypts it for the tallier. It also generates the visual shares $P(x) = (v_1, v_2)$ that yield the visual description $v_0 = v_1 \vee v_2$, with $v_0 \in D_x$. The PC prepares a proof of compatibility of the visual description, the shares and the vote: It commits through Pedersen commitments $C_{0_j}, C_{1_j}, C_{2_j}$ to each bit $j = 0, \dots, 2k - 1$ of v_0, v_1, v_2 respectively, and proves the validity of bit commitments. Moreover it proves that the splitting is valid, i.e. that for each bit j $v_{0_j} = v_{1_j} \vee v_{2_j}$. To do so we observe that the latter is true if and only if the commitment $C_j = (C_{1_j})(C_{2_j})^2(C_{0_j})^3$ hides a value in $\{0, 4, 5, 6\}$. In addition the PC proves that $v_0 = (v_{0_{2k-1}} v_{0_{2k-2}} \dots v_{0_1} v_{0_0})$ is a valid visual description of x, i.e. $v_0 \in D_x$. That is for each bit b_j, $j = 0, \dots, k-1$ in the binary encoding of x, it holds that $b_j = v_{0_{2j+1}} \oplus v_{0_{2j}} = v_{0_{2j+1}} + v_{0_{2j}} \bmod 2$. Since we have proved that all values v_{0_j} are bits and for all $j = 0, \dots, k-1$ $(v_{0_{2j+1}} v_{0_{2j}}) \in \cup_{x \in M^*} D^*_x$, i.e. $(v_{0_{2j+1}} v_{0_{2j}}) \in \{01, 10, 11\}$, the latter relation is equivalent to proving that $b_j = 2 - v_{0_{2j+1}} - v_{0_{2j}}$. Thus we prove knowledge of the value x in the encryption E_x and then prove that the commitment $\prod_{j=0}^{k-1} (\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{2^j}$ also hides the same value x. The full interactive proof of knowledge is denoted as $\pi' = PK(x, \rho, \{(v_{i_j}, r_{i_j})_{i=0,1,2}\}_{j=0}^{2k-1} \mid \{(C_{i_j} = g^{v_{i_j}} h^{r_{i_j}})_{i=0,1,2}\}_{j=0}^{2k-1} \wedge (\{(v_{i_j})_{i=0,1,2}\}_{j=0}^{2k-1} \in \{0,1\}) \wedge (v_{1_j} + 2v_{2_j} + 3v_{0_j} \in \{0, 4, 5, 6\})_{j=0}^{2k-1} \wedge E_x = g^x pk_t^\rho \wedge (x = \sum_{j=0}^{k-1} (2 - v_{0_{2j+1}} - v_{0_{2j}}) 2^j))$. Due to lack of space its full description is given in appendix C. For our protocol we will use the non-interactive version by the Fiat-Shamir heuristic. Its security follows from the results of [6], however we provide a proof in appendix D.

Each server S_i $(i = 1, 2)$ needs to verify the proof π' and the validity of the share v_i. The servers store the shares v_i for all $v_i \in \Lambda$, as well as their corresponding images that will be sent back to the voters. Let $\tilde{v_i}$ be the number whose binary encoding is the bitstring $v_i = (v_{i_{2k-1}} v_{i_{2k-2}} \dots v_{i_1} v_{i_0}) \in \Lambda$, i.e. $\tilde{v_i} = \sum_{j=0}^{2k-1} v_{i_j} 2^j$. Each server S_i stores a database of all valid shares $v_i \in \Lambda$ and their images, indexed by the corresponding value $\tilde{v_i}$. By this construction we do not need to open all bit commitments $\{C_{i_j} = g^{v_{i_j}} h^{r_{i_j}}\}_{j=0}^{2k-1}$ of the bits of v_i to server S_i, in order to verify the validity of the share. Instead the PC can open to server S_i a single commitment $C'_i = \prod_{j=0}^{2k-1} (C_{i_j})^{2^j}$ as $g^{\tilde{v_i}} h^{\rho_{\tilde{v_i}}}$ with $\rho_{\tilde{v_i}} = \sum_{j=0}^{2k-1} r_{i_j} 2^j$, so that the server can verify that the bit commitments C_{i_j} correspond to a valid share v_i. We note that $\tilde{v_i}$ needs to be in $\mathbb{Z}_q$ so we should have $2^{2k} - 1 < q$. If all verifications are successful the server S_i sends the image indexed by $\tilde{v_i}$ to the voter through the untappable channel.

The Visual Vote Verification Protocol: Let M be the set of m candidates, $k = \lfloor \log_2(m-1) \rfloor + 1$, $(pk_{S_i}, sk_{S_i}), (pk_t, sk_t)$ be the public/secret key pairs of server S_i $(i = 1, 2)$ and the tallier respectively, (sk_V, vk_V) and (sg_{S_i}, vk_{S_i}) be the signing/verification key pairs of voter V and server S_i, h the commitment public key and $\langle p, q, g \rangle$ the system parameters.

- The voter V votes for candidate x through her PC and waits for the images from the voting servers. Upon receiving them she verifies that their overlaying is a correct shape description for candidate x.
- The PC $(sk_V, pk_t, pk_{S_1}, pk_{S_2})$ on input x by the voter:
  1. Generates a valid visual splitting $(v_1, v_2) \leftarrow P(x)$ for candidate x and its visual representation $v_0 = v_1 \vee v_2 \in D_x$.
  2. Picks $r, \rho \leftarrow \mathbb{Z}_q$ and encrypts the vote $E_t = (E_x, E_\rho) = Enc_{pk_t}(g^x, \rho)$.
  3. For each $j = 0, \dots, 2k-1$ commits to the j-th bit of v_0, v_1, v_2 as $C_{i_j} = g^{v_{i_j}} h^{r_{i_j}}$ for $i = 0, 1, 2$ and $\{r_{i_j}\}_{i=0,1,2}^{j=0,\dots,2k-1} \leftarrow \mathbb{Z}_q$.

4. Let $\tilde{v_1}, \tilde{v_2} \in \mathbb{Z}_q$ be the values whose binary representation is $v_1, v_2 \in \{0,1\}^{2k}$. Then for $i = 1, 2$ the PC encrypts the opening randomness $\rho_{\tilde{v_i}} = \sum_{j=0}^{2k-1} r_{i_j} 2^j$ as $R_{v_i} = Enc_{pk_{S_i}}(\rho_{\tilde{v_i}}, r'_i)$, for $r'_i \leftarrow \mathbb{Z}_q$ and the values $\tilde{v_i}$ as $e_{v_i} = Enc_{pk_{S_i}}(\tilde{v_i}, r_i)$, for $r_i \leftarrow \mathbb{Z}_q$.
5. Prepares the non-interactive proof of knowledge $\pi' = PK(x, \rho, \{(v_{i_j}, r_{i_j})_{i=0,1,2}\}_{j=0}^{2k-1} \mid ((C_{i_j} = g^{v_{i_j}} h^{r_{i_j}})_{i=0,1,2})_{j=0}^{2k-1} \wedge (\{(v_{i_j})_{i=0,1,2}\}_{j=0}^{2k-1} \in \{0,1\}) \wedge (v_{1_j} + 2v_{2_j} + 3v_{0_j} \in \{0,4,5,6\})_{j=0}^{2k-1} \wedge (E_x = g^x pk_t^{\rho}) \wedge (x = \sum_{j=0}^{k-1}(2 - v_{0_{2j+1}} - v_{0_{2j}})2^j)$.
6. Signs the encrypted message and the proof.
7. Sends to server S_i, for $i = 1, 2$, $D_i = (e_{v_i}, R_{v_i})$ and posts $B = (E_t, \{(C_{i_j})_{i=0,1,2}\ \}_{j=0}^{2k-1}, \pi', sing_{sk_V}(E_t, \pi'), V)$ to the broadcast channel.

– The Server S_i $(sg_{S_i}, sk_{S_i}, vk_V, \langle \tilde{v_i}, v_i, \text{image}(v_i)\rangle_{v_i \in \Lambda})$ on input D_i, B performs the following tests. If any step fails it declares a forgery and stops:
  1. Verifies the voter's signature and proof π'.
  2. Decrypts e_{v_i}, R_{v_i} to obtain $\tilde{v_i}, \rho_{v_i}$, checks that $\tilde{v_i}$ is a valid entry in the database $\langle \tilde{v_i}, v_i, \text{image}(v_i)\rangle_{v_i \in \Lambda}$ and checks that $g^{\tilde{v_i}} h^{\rho_{\tilde{v_i}}} = \prod_{j=0}^{2k-1}(C_{i_j})^{2^j}$.
  3. Sends the corresponding image of $v_i \in \Lambda$ to the voter through the untappable channel. It signs and stores the vote $sign_{sg_{S_i}}(E_t)$.

– The Tallier: Obtains from a server the votes E_t and runs a suitable protocol.

Security. *Cast as intended:* Let x be the vote submitted by the voter and x' the forged vote such that $Dec_{sk_t}(E_t) = x'$. In order for x' to create a valid receipt v_0 for x it should hold that $v_0 \in D_x$, which implies that for all $j = 0, \dots, k-1$ $(v_{0_{2j+1}} v_{0_{2j}}) \in D^*_{b_j}$ where b_j is the j-th bit of x, and thus $v_{0_{2j+1}} \oplus v_{0_{2j}} = b_j$. It follows that $\sum_{j=0}^{k-1}(v_{0_{2j+1}} \oplus v_{0_{2j}})2^j = \sum_{j=0}^{k-1} b_j 2^j = x$. Since proof π' guarantees that E_t and $\sum_{j=0}^{k-1}(v_{0_{2j+1}} \oplus v_{0_{2j}})2^j$ hide the same value, we have that $x = x'$.

(1,2)-Vote secrecy: Without loss of generaltiy let the attacker control server S_1 and let $x_0, x_1 \in \mathbb{Z}_m$ be the candidates chosen by the attacker. The challenger randomly selects and encrypts x_b ($b \in \{0,1\}$) and produces the shares $(v_1, v_2) \in P(x_b)$. The public commitments and encryptions do not reveal information to the attacker, who neither extracts information from v_1 since $Prob[(v_1, \#) \in P(x_0)] = Prob[(v_1, \#) \in P(x_1)]$ from the $(1,2)$-resilience property.

Complexity. The PC needs 10 exponentiations for encryptions and commitments, $12k$ exponentiations for bit commitments and $44k+3$ exponentiations for generating the proof π', a total of $56(\lfloor \log_2(m-1) \rfloor + 1) + 15$ online exponentiations and one signing. Each server S_i does 4 exponentiations for decryptions and commitment openings, $56k+5$ for verifying π' and k for checking the compatibility of the share and its bits, a total of $57(\lfloor \log_2(m-1) \rfloor + 1) + 9$ exponentiations, one signing and one signature verification.

Extensions. The 2-VSSD scheme we presented can be extended to n-VSSD in a number of possible ways. We leave it as future work to determine which ones might be more suitable for human verifiability. We also leave it as open question to develop the case where resiliency is achieved for $t > 1$ but we conjecture that it is possible to obtain a general n-VSSD ($n > 2$) with (t, n)-resilience for $t > 1$ using techniques that were developed for *colored* visual secret sharing [25].

References

1. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012)
2. Boneh, D., Golle, P.: Almost entirely correct mixing with applications to voting. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, pp. 68–77. ACM, New York (2002)
3. Chaabouni, R., Lipmaa, H., Shelat, A.: Additive combinatorics and discrete logarithm based range protocols. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 336–351. Springer, Heidelberg (2010)
4. Chaum, D.: Surevote. International patent WO 01/55940 A1 (2001)
5. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. IEEE Security & Privacy 2(1), 38–47 (2004)
6. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs 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)
7. 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)
8. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
9. Gjøsteen, K.: Analysis internet voting protocol. Technical Report (2010), `http://www.regjeringen.no`
10. Gjøsteen, K.: Analysis of an internet voting protocol. IACR Cryptology ePrint Archive 2010:380 (2010)
11. Gjøsteen, K.: The norwegian internet voting protocol. In: Kiayias, A., Lipmaa, H. (eds.) VoteID 2011. LNCS, vol. 7187, pp. 1–18. Springer, Heidelberg (2012)
12. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2003)
13. Heather, J., Ryan, P.Y.A., Teague, V.: Pretty good democracy for more expressive voting schemes. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 405–423. Springer, Heidelberg (2010)
14. Heiberg, S., Lipmaa, H., van Laenen, F.: On E-vote integrity in the case of malicious voter computers. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 373–388. Springer, Heidelberg (2010)
15. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Proceedings of the 11th USENIX Security Symposium, pp. 339–353. USENIX Association, Berkeley (2002)
16. Lipmaa, H.: Two simple code-verification voting protocols. IACR Cryptology ePrint Archive, 2011:317 (2011)
17. Lipmaa, H., Asokan, N., Niemi, V.: Secure vickrey auctions without threshold trust. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 87–101. Springer, Heidelberg (2003)

18. Mao, W.: Guaranteed correct sharing of integer factorization with off-line shareholders. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 60–71. Springer, Heidelberg (1998)
19. Naor, M., Shamir, A.: Visual cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)
20. Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001, pp. 116–125. ACM, New York (2001)
21. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
22. Ryan, P.Y.A.: A variant of the chaum voter-verifiable scheme. In: WITS, pp. 81–88 (2005)
23. Ryan, P.Y.A., Teague, V.: Pretty good democracy. In: Christianson, B., Malcolm, J.A., Matyáš, V., Roe, M. (eds.) Security Protocols 2009. LNCS, vol. 7028, pp. 111–130. Springer, Heidelberg (2013)
24. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
25. Verheul, E.R., Van Tilborg, H.C.A.: Constructions and properties of k out of nvisual secret sharing schemes. Des. Codes Cryptography 11(2), 179–196 (1997)

Appendix A

We provide an example of the visual description of a scheme with 4 candidates along with the relevant splittings. The visual alphabet for this representation is $\Lambda = \{01, 10\}^2$.

Message	Shape	D_x	$P(x)$
00	Two full circles	● ●	(◐ ◐, ◑ ◑) (◐ ◑, ◑ ◐) (◑ ◐, ◐ ◑) (◑ ◑, ◐ ◐)
01	Full circle followed by half circle	● ◐ ● ◑	(◐ ◐, ◑ ◐) (◑ ◐, ◐ ◐) (◐ ◑, ◑ ◑) (◑ ◑, ◐ ◑)
10	Half circle followed by full circle	◐ ● ◑ ●	(◐ ◐, ◐ ◑) (◐ ◑, ◐ ◐) (◑ ◑, ◑ ◐) (◑ ◐, ◑ ◑)
11	Two half circles	◐ ◐, ◐ ◑ ◑ ◐, ◑ ◑	(◐ ◐, ◐ ◐) (◐ ◑, ◐ ◑) (◑ ◐, ◑ ◐) (◑ ◑, ◑ ◑)

Fig. 2. Visual descriptions and splittings of 4 candidates

The depiction of the splittings along with the relevant shares exhibits the solvability property, while for a randomly selected share w in Λ , say $w =$ ◐◐, it holds that $Prob[($◐◐$, \#) \in P(00)] = Prob[($◐◐$, \#) \in P(01)] = Prob[($◐◐$, \#) \in P(10)] = Prob[($◐◐$, \#) \in P(11)] = 1/4$, satisfying $(1, 2)$-resilience. The same holds for the rest of the cases with $w =$ ◐◑, $w =$ ◑◐, $w =$ ◑◑ and the symmetric case $(\#, w)$.

Appendix B

Proof of Proposition 1

Proof. For the message 0 the description set is $D_0 = \{11\}$ and possible splittings are $P(0) \in \{(10, 01), (01, 10)\}$, and analogously for the message 1 $D_1 = \{01, 10\}$ and $P(1) \in \{(01, 01), (10, 10)\}$.

Solvability: For message 0 we have $10 \vee 01 = 11 \in D_0$ and $01 \vee 10 = 11 \in D_0$. Similarly for message 1, we have $01 \vee 01 = 01 \in D_1$ and $10 \vee 10 = 10 \in D_1$.

(1,2)-Resilience: The 2-tuples with one known and one unknown element in Λ we need to consider are $(01, \#), (10, \#), (\#, 01), (\#, 10)$. The function P

outputs one of the possible valid splittings uniformly at random and thus we have $Prob[(01,\#) \in P(0)] = Prob[(01,\#) \in P(1)] = \frac{1}{2}$, $Prob[(10,\#) \in P(0)] = Prob[(10,\#) \in P(1)] = \frac{1}{2}$ and $Prob[(\#,01) \in P(0)] = Prob[(\#,01) \in P(1)] = \frac{1}{2}$, $Prob[(\#,10) \in P(0)] = Prob[(\#,10) \in P(1)] = \frac{1}{2}$.

Proof of Proposition 2

Proof. **Solvability:** Let $(a_{k-1} \dots a_0, a'_{k-1} \dots a'_0)$ be a splitting in $P(x)$. From the construction of P for all $j = 0, \dots, k-1$ $(a_j, a'_j) \in P^*(b_j)$ where b_j is the j-th bit of x. Then from proposition 1 for all $j = 0, \dots, k-1$ it holds that $d_j = a_j \vee a'_j$ is in $D^*_{b_j}$, which implies that $(d_{k-1} \dots d_0) \in D_x$ as requested.

(1,2)-Resilience: Let $(a_{k-1} \dots a_0, \#_k)$ denote the tuple with exactly one known element $(a_{k-1} \dots a_0)$ in $\Lambda = \{01, 10\}^k$. Thus for all $j = 0, \dots, k-1$ $a_j = 01$ or $a_j = 10$, i.e. $a_j \in \Lambda^*$. Since the splitting function $P(x)$ handles each bit b_j of x independently from the others, we have that for all $x \in M$ $Prob[(a_{k-1} \dots a_0, \#_k) \in P(x)] = Prob[(a_{k-1}, \#) \in P^*(b_{k-1})] \times \dots \times Prob[(a_0, \#) \in P^*(b_0)]$, where $b_j \in M^*$ is the j-th bit of x. Since from proposition 1 we have that for all $a_j \in \Lambda^*$ $Prob[(a_j, \#) \in P^*(0)] = Prob[(a_j, \#) \in P^*(1)] = \frac{1}{2}$ we conclude that for all $x \in M$ $Prob[(a_{k-1} \dots, a_0, \#_k) \in P(x)] = (\frac{1}{2})^k$ as requested.
The case of $(\#_k, a_{k-1} \dots a_0)$ is symmetrical.

Appendix C

The proof $\boldsymbol{\pi'}$: Public Input: $\langle p, q, g \rangle$, h, pk_t, m, $k = \lfloor \log_2(m-1) \rfloor + 1$, $E_t = (E_x, E_\rho) = (g^x pk_t^\rho, g^\rho)$, $\{C_{1_j} = g^{v_{1j}} h^{r_{1j}}, C_{2_j} = g^{v_{2j}} h^{r_{2j}}, C_{0_j} = g^{v_{0j}} h^{r_{0j}}\}_{j=0}^{2k-1}$. Let $d_1 = 1, d_2 = g^4, d_3 = g^5, d_4 = g^6$.

Prover's Input: $x, \rho, \{v_{1_j}, r_{1_j}, v_{2_j}, r_{2_j}, v_{0_j}, r_{0_j}\}_{j=0}^{2k-1}$.

1. The Prover:
   (a) *Bit proof:* For $j = 0, \dots, 2k-1$:
      i. For $i = 0, 1, 2$:
         – If $v_{i_j} = 0$ it picks $w_{ij}, c_{2ij}, \rho_{2ij} \leftarrow \mathbb{Z}_q$ and sets $y_{1ij} = h^{w_{ij}}$, $y_{2ij} = h^{\rho_{2ij}} (C_{i_j}/g)^{-c_{2ij}}$.
         – Else if $v_{1_j} = 1$ it picks $w_{ij}, c_{1ij}, \rho_{1ij} \leftarrow \mathbb{Z}_q$ and sets $y_{1ij} = h^{\rho_{1ij}} (C_{i_j})^{-c_{1ij}}$, $y_{2ij} = h^{w_{ij}}$
   (b) *OR proof:* For $j = 0, \dots, 2k-1$:
      i. If $1v_{1_j} + 2v_{2_j} + 3v_{0_j} = 0$ (case $0 \vee 0 = 0$) set $t = 1$. If $1v_{1_j} + 2v_{2_j} + 3v_{0_j} = 4$ (case $1 \vee 0 = 1$) set $t = 2$. If $1v_{1_j} + 2v_{2_j} + 3v_{0_j} = 5$ (case $0 \vee 1 = 1$) set $t = 3$. If $1v_{1_j} + 2v_{2_j} + 3v_{0_j} = 6$ (case $1 \vee 1 = 1$) set $t = 4$.
      ii. It picks $w_{tj}, \{c_{\lambda j}, \rho_{\lambda j}\}_{\lambda=1,2,3,4}^{\lambda \neq t} \leftarrow \mathbb{Z}_q$ and sets $y_{tj} = h^{w_{tj}}$ and $\{y_{\lambda j} = h^{\rho_{\lambda j}} (\frac{(C_{1_j})(C_{2_j})^2 (C_{0_j})^3}{d_\lambda})^{-c_{\lambda j}}\}_{\lambda=1,2,3,4}^{\lambda \neq t}$.
   (c) It picks $w, \rho_1, \rho_2 \leftarrow \mathbb{Z}_q$ and sets $y_1 = g^w pk_t^{\rho_1}$, $y_2 = g^w h^{\rho_2}$.
   (d) It sends (y_1, y_2), $(y_{1j}, y_{2j}, y_{3j}, y_{4j})_{j=0}^{2k-1}$,$((y_{1ij}, y_{2ij})_{j=0}^{2k-1})_{i=0,1,2}$ to the Verifier.
2. The Verifier picks $\mathbf{c} \leftarrow \mathbb{Z}_q$ and sends it to the Prover.

3. The Prover:
   (a) *Bit proof:* For $j = 0, \ldots, 2k-1$:
      – For $i = 0, 1, 2$:
         i. If $v_{i_j} = 0$ it sets $c_{1ij} = \mathbf{c} - c_{2ij}$, $\rho_{1ij} = w_{ij} + c_{1ij} r_{i_j}$.
         ii. Else if $v_{i_j} = 1$ it sets $c_{2ij} = \mathbf{c} - c_{1ij}$, $\rho_{2ij} = w_{ij} + c_{2ij} r_{i_j}$.
   (b) *OR proof:* For $j = 0, \ldots, 2k-1$:
      – It sets $c_{tj} = \mathbf{c} - (\sum_{\lambda=1,2,3,4}^{\lambda \neq t} c_{\lambda j})$ and $\rho_{tj} = w_{tj} + c_{tj}(r_{1_j} + 2r_{2_j} + 3r_{0_j})$.
   (c) It sets $s = w + \mathbf{c}x$, $\rho_1' = \rho_1 + \mathbf{c}\rho$, $\rho_2' = \rho_2 - \mathbf{c}(\sum_{j=0}^{k-1}(r_{0_{2j+1}} + r_{0_{2j}})2^j)$.
   (d) It sends to the Verifier (s, ρ_1', ρ_2') , $((c_{\lambda j}, \rho_{\lambda_j})_{\lambda=1,2,3,4})_{j=0}^{2k-1}$ and $((c_{1ij}, c_{2ij}, \rho_{1ij}, \rho_{2ij})_{j=0}^{2k-1})_{i=0,1,2}$.
4. The Verifier accepts if all the following tests succeed, otherwise it rejects:
   (a) For $j = 0, \ldots, 2k-1$:
      i. *Bit proof:* For $i = 0, 1, 2$: $\mathbf{c} = c_{1ij} + c_{2ij}$, $h^{\rho_{1ij}} = y_{1ij}(C_{i_j})^{c_{1ij}}$ and $h^{\rho_{2ij}} = y_{2ij}(C_{i_j}/g)^{c_{2ij}}$.
      ii. *OR proof:* $\mathbf{c} = \sum_{\lambda=1}^{4} c_{\lambda j}$ and for $\lambda = 1, 2, 3, 4$:
      $h^{\rho_{\lambda j}} = y_{\lambda j}(\frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda})^{c_{\lambda j}}$.
   (b) $g^s pk_t^{\rho_1'} = y_1(E_x)^{\mathbf{c}}$ and $g^s h^{\rho_2'} = y_2(\prod_{j=0}^{k-1}(\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{2^j})^{\mathbf{c}}$.

Appendix D

Security Proof for Proof of Knowledge π

Proof. **Completeness:** Executing the protocol with an honest prover and a honest verifier, we have that in step 4 condition (b) holds since $\prod_{j=0}^{k-1} \mathcal{E}_j^{H_j} = \prod_{j=0}^{k-1}(g^{\mu_j} pk_t^{z_j})^{H_j} = g^{\sum_{j=0}^{k-1} \mu_j H_j} pk_t^{\sum_{j=0}^{k-1} z_j H_j} = g^x pk_t^r = E_x$.
Let i^* be the value such that $\sum_{i=1}^n x_i = x + (i^* - 1)u$. Then condition (c) holds since $\sum_{i=1}^n c_i = c$ and for all $i = 1, \ldots, n$ with $i \neq i^*$ we have that:

$$a_i(E_x)^{c_i} = ((E_x)^{-c_i} g^{s_i} pk_t^{\rho_{1i}'})(E_x)^{c_i} = g^{s_i} pkt^{\rho_{1i}'}$$

and

$$b_i(\prod_{l=1}^n C_l/g^{(i-1)u})^{c_i} = (\prod_{l=1}^n C_l/g^{(i-1)u})^{-c_i} g^{s_i} h^{\rho_{2i}'}(\prod_{l=1}^n C_l/g^{(i-1)u})^{c_i} = g^{s_i} h^{\rho_{2i}'}.$$

For i^* it holds that:

$$a_{i^*}(E_x)^{c_{i^*}} = (g^w pk_t^{\rho_a})(g^x pk_t^r)^{c_{i^*}} = g^{w+xc_{i^*}} pk_t^{\rho_a + rc_{i^*}} = g^{s_{i^*}} pk_t^{\rho_{1i^*}'}$$

and

$$b_{i^*}(\prod_{l=1}^n C_l/g^{(i^*-1)u})^{c_{i^*}} = g^w h^{\rho_b}(\prod_{l=1}^n g^{x_l} h^{r_l}/g^{(i^*-1)u})^{c_{i^*}} =$$

$$= g^w h^{\rho_b}(g^{\sum_{l=1}^n x_l - (i^*-1)u} h^{\sum_{l=1}^n r_l})^{c_{i^*}} = g^{w + (\sum_{l=1}^n x_l - (i^*-1)u)c_{i^*}} h^{\rho_b + (\sum_{l=1}^n r_l)c_{i^*}} =$$

$$= g^{w+xc_{i*}} h^{\rho_b+(\sum_{l=1}^{n} r_l)c_{i*}} = g^{s_{i*}} h^{\rho'_{2i*}}.$$

Finally condition (a) holds as for all $j = 0, \dots, k-1$: $c = c_{1j} + c_{2j}$, and if $\mathcal{E}_j = g^0 pk_t^{z_j}$, we have that:

$$y_{1j}(\mathcal{E}_j)^{c_{1j}} = pk_t^{w_j}(pk_t^{z_j})^{c_{1j}} = pk_t^{w_j+c_{1j}z_j} = pkt^{\rho_{1j}}$$

and

$$y_{2j}(\mathcal{E}_j/g)^{c_{2j}} = pk_t^{\rho_{2j}}(\mathcal{E}_j/g)^{-c_{2j}}(\mathcal{E}_j/g)^{c_{2j}} = pk_t^{\rho_{2j}}.$$

Otherwise if $\mathcal{E}_j = g^1 pk_t^{z_j}$ then:

$$y_{1j}(\mathcal{E}_j)^{c_{1j}} = pk_t^{\rho_{1j}}(\mathcal{E}_j)^{-c_{1j}}(\mathcal{E}_j)^{c_{1j}} = pk_t^{\rho_{1j}}$$

and

$$y_{2j}(\mathcal{E}_j/g)^{c_{2j}} = pk_t^{w_j}(g^1 pk_t^{z_j}/g^1)^{c_{2j}} = pk_t^{w_j+z_jc_{2j}} = pk_t^{\rho_{2j}}.$$

Special Soundness: Let

$$\langle A, c, B\rangle =$$

$$= \langle(\{a_i, b_i\}_{i=1}^{n}, \{\mathcal{E}_j, y_{1j}, y_{2j}\}_{j=0}^{k-1}), c, (\{c_i, s_i, \rho'_{1i}, \rho'_{2i}\}_{i=1}^{n}, \{c_{1j}, c_{2j}, \rho_{1j}, \rho_{2j}\}_{j=0}^{k-1})\rangle$$

and

$$\langle A, \tilde{c}, \tilde{B}\rangle =$$

$$= \langle(\{a_i, b_i\}_{i=1}^{n}, \{\mathcal{E}_j, y_{1j}, y_{2j}\}_{j=0}^{k-1}), \tilde{c}, (\{\tilde{c_i}, \tilde{s_i}, \tilde{\rho'_{1i}}, \tilde{\rho'_{2i}}\}_{i=1}^{n}, \{\tilde{c_{1j}}, \tilde{c_{2j}}, \tilde{\rho_{1j}}, \tilde{\rho_{2j}}\}_{j=0}^{k-1})\rangle$$

be two accepting communication transcripts for the same first message A with $c \neq \tilde{c}$.

Since both transcripts are accepting we have that from condition (c) for $i = 1, \dots, n$:

$$g^{s_i} pk_t^{\rho'_{1i}} = a_i(E_x)^{c_i} \quad \text{and} \quad g^{\tilde{s_i}} pk_t^{\tilde{\rho'_{1i}}} = a_i(E_x)^{\tilde{c_i}}$$

and

$$g^{s_i} h^{\rho'_{2i}} = b_i(\prod_{l=1}^{n} C_l/g^{(i-1)u})^{c_i} \quad \text{and} \quad g^{\tilde{s_i}} h^{\tilde{\rho'_{2i}}} = b_i(\prod_{l=1}^{n} C_l/g^{(i-1)u})^{\tilde{c_i}}.$$

Then since $c \neq \tilde{c}$ for some $i \in \{1, \dots, n\}$ there are $c_i \neq \tilde{c_i}$ so that we have:

$$\frac{g^{s_i} pk_t^{\rho'_{1i}}}{g^{\tilde{s_i}} pk_t^{\tilde{\rho'_{1i}}}} = (E_x)^{c_i-\tilde{c_i}} \Leftrightarrow g^{\frac{s_i-\tilde{s_i}}{c_i-\tilde{c_i}}} pk_t^{\frac{\rho'_{1i}-\tilde{\rho'_{1j}}}{c_i-\tilde{c_i}}} = E_x$$

so we can extract a valid opening for $E_x = g^{w_i} pk_t^{v_{1i}}$ with $w_i = \frac{s_i-\tilde{s_i}}{c_i-\tilde{c_i}}$ and $v_{1i} = \frac{\rho'_{1i}-\tilde{\rho'_{1j}}}{c_i-\tilde{c_i}}$.
In addition since:

$$\frac{g^{s_i} pk_t^{\rho'_{2i}}}{g^{\tilde{s_i}} pk_t^{\tilde{\rho'_{2i}}}} = (\prod_{l=1}^{n} C_l/g^{(i-1)u})^{c_i-\tilde{c_i}} \Leftrightarrow g^{\frac{s_i-\tilde{s_i}}{c_i-\tilde{c_i}}} pk_t^{\frac{\rho'_{2i}-\tilde{\rho'_{2i}}}{c_i-\tilde{c_i}}} = \prod_{l=1}^{n} C_l/g^{(i-1)u}$$

we also extract a valid opening for $\prod_{l=1}^{n} C_l/g^{(i-1)u} = g^{w_i} pk_t^{v_{2i}}$ with w_i and $v_{2i} = \frac{\rho'_{2i}-\tilde{\rho'_{2i}}}{c_i-\tilde{c_i}}$.

From condition (a) we have that since $c \neq \tilde{c}$ for all $j = 0, \ldots, k-1$, either $c_{1j} \neq \tilde{c_{1j}}$ or $c_{2j} \neq \tilde{c_{2j}}$. We also have that:

$$pk_t^{\rho_{1j}} = y_{1j}(\mathcal{E}_j)^{c_{1j}} \quad \text{and} \quad pk_t^{\tilde{\rho_{1j}}} = y_{1j}(\mathcal{E}_j)^{\tilde{c_{1j}}}$$

and

$$pk_t^{\rho_{2j}} = y_{2j}(\mathcal{E}_j/g)^{c_{2j}} \quad \text{and} \quad pk_t^{\tilde{\rho_{2j}}} = y_{2j}(\mathcal{E}_j/g)^{\tilde{c_{2j}}}.$$

Then in the case that $c_{1j} \neq \tilde{c_{1j}}$ we have that:

$$\frac{pk_t^{\rho_{1j}}}{pk_t^{\tilde{\rho_{1j}}}} = \frac{(\mathcal{E}_j)^{c_{1j}}}{(\mathcal{E}_j)^{\tilde{c_{1j}}}} \Leftrightarrow pk_t^{\frac{\rho_{1j}-\tilde{\rho_{1j}}}{c_{1j}-\tilde{c_{1j}}}} = \mathcal{E}_j$$

so we can extract a valid opening for $\mathcal{E}_j = pk_t^{z_{1j}}$ with $z_{1j} = \frac{\rho_{1j}-\tilde{\rho_{1j}}}{c_{1j}-\tilde{c_{1j}}}$. Similarly in the case that $c_{2j} \neq \tilde{c_{2j}}$ from $\frac{pk_t^{\rho_{2j}}}{pk_t^{\tilde{\rho_{2j}}}} = \frac{(\mathcal{E}_j/g)^{c_{2j}}}{(\mathcal{E}_j/g)^{\tilde{c_{2j}}}}$ we extract a valid opening for $\mathcal{E}_j/g = pk_t^{z_{2j}}$ with $z_{2j} = \frac{\rho_{2j}-\tilde{\rho_{2j}}}{c_{2j}-\tilde{c_{2j}}}$.

HV Zero Knowledge: We can create a simulator for the prover given the public input $G, q, g, pk_t, h, E_x, \{C_i\}_{i=1}^n$ as follows: We randomly pick $c \leftarrow \mathbb{Z}_q$ and for $i = 1, \ldots n$ we pick $c_{1i}, s_i, \rho'_{1i}, \rho'_{2i} \leftarrow \mathbb{Z}_q$ and set $c_{2i} = c - c_{1i} \mod q$. For $j = 0, \ldots, k-1$ we pick $c_{1j}, \rho_{1j}, \rho_{2j} \leftarrow \mathbb{Z}_q$ and set $c_{2j} = c - c_{1j} \mod q$. We fix the second and the third message of the communication protocol to be $\langle c, (\{c_i, s_i, \rho'_{1i}, \rho'_{2i}\}_{i=1}^n, \{c_{1j}, c_{2j}, \rho_{1j}, \rho_{2j}\}_{j=0}^{k-1}) \rangle$. Then we set the first message A to be $A = (\{(E_x)^{-c_i} g^{s_i} pk_t^{\rho'_{1i}}, (\prod_{l=1}^n C_l/g^{(i-1)u})^{-c_i} g^{s_i} h^{\rho'_{2i}}\}_{i=1}^n$, $\{\mathcal{E}_j, pk_t^{\rho_{1j}}(\mathcal{E}_j)^{-c_{1j}}, pk_t^{\rho_{2j}}(\mathcal{E}_j/g)^{-c_{2j}}\}_{j=0}^{k-1})$.

Security Proof for Proof of Knowledge π'

Proof. **Completeness:** Executing the protocol with an honest prover and a honest verifier, we have that in step 4 condition (b) holds since

$$y_1(E_x)^{\mathbf{c}} = g^w pk_t^{\rho_1}(g^x pk_t^{\rho})^{\mathbf{c}} = g^{w+\mathbf{c}x} pk_t^{\rho_1+\mathbf{c}\rho} = g^s pk_t^{\rho'_1}$$

and

$$y_2(\prod_{j=0}^{k-1}(\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{2^j})^{\mathbf{c}} =$$

$$= g^w h^{\rho_2}(g^{(\sum_{j=0}^{k-1}(2-v_{0_{2j+1}}-v_{0_{2j}})2^j)} h^{(-\sum_{j=0}^{k-1}(r_{0_{2j+1}}+r_{0_{2j}})2^j)})^{\mathbf{c}} = g^{w+\mathbf{c}x} h^{\rho'_2} = g^s h^{\rho'_2}.$$

Condition (ai) holds as for all $j = 0, \ldots, 2k-1$ and for all $i = 0, 1, 2$: $\mathbf{c} = c_{1ij} + c_{2ij}$, and if $C_{i_j} = g^0 h^{r_{i_j}}$, we have that:

$$y_{1ij}(C_{i_j})^{c_{1ij}} = h^{w_{ij}}(h^{r_{i_j}})^{c_{1ij}} = h^{\rho_{1ij}}$$

and

$$y_{2ij}(C_{i_j}/g)^{c_{2ij}} = (h^{\rho_{2ij}}(C_{i_j}/g)^{-c_{2ij}})(C_{i_j}/g)^{c_{2ij}} = h^{\rho_{2ij}}.$$

Otherwise if $C_{i_j} = g^1 h^{r_{i_j}}$ we have that:

$$y_{1ij}(C_{i_j})^{c_{1ij}} = (h^{\rho_{1ij}}(C_{i_j})^{-c_{1ij}})(C_{i_j})^{c_{1ij}} = h^{\rho_{1ij}}$$

and

$$y_{2ij}(C_{i_j}/g)^{c_{2ij}} = h^{w_{ij}}(gh^{r_{i_j}}/g)^{c_{2ij}} = h^{w_{ij}+c_{2ij}r_{i_j}} = h^{\rho_{2ij}}.$$

Regarding condition (aii), for $j = 0, \dots, 2k-1$, we present the case for general $t \in \{1,2,3,4\}$ with $d_t = g^\alpha$, i.e. $(v_{1_j}) + 2(v_{2_j}) + 3(v_{0_j}) = \alpha$, with $\alpha \in \{0,4,5,6\}$. The condition holds as $\mathbf{c} = c_{1j} + c_{2j} + c_{3j} + c_{4j}$ and for $t : y_{tj}(\frac{(C_{1_j})(C_{2_j})^2(C_{3_j})^3}{d_t})^{c_{tj}} =$

$$= h^{w_{tj}}(\frac{g^{(1v_{1_j}+2v_{2_j}+3v_{0_j})}h^{(1r_{1_j}+2r_{2_j}+3r_{0_j})}}{g^\alpha})^{c_{tj}} =$$

$$= h^{w_{tj}}(\frac{g^\alpha h^{(1r_{1_j}+2r_{2_j}+3r_{0_j})}}{g^\alpha})^{c_{tj}} = h^{w_{tj}+c_{1j}(1r_{1_j}+2r_{2_j}+3r_{0_j})} = h^{\rho_{1j}}.$$

For $\lambda \neq t$ the proof is simulated and thus $y_{\lambda j}(\frac{(C_{1_j})(C_{2_j})^2(C_{3_j})^3}{d_\lambda})^{c_{\lambda j}} =$

$$= (h^{\rho_{\lambda j}}(\frac{(C_{1_j})(C_{2_j})^2(C_{3_j})^3}{d_\lambda})^{-c_{\lambda j}})(\frac{(C_{1_j})(C_{2_j})^2(C_{3_j})^3}{d_\lambda})^{c_{\lambda j}} = h^{\rho_{\lambda j}}.$$

Special Soundness:

Let $\langle A, c, B\rangle$, $\langle A, \tilde{c}, \tilde{B}\rangle$ be two accepting communication transcripts for the same first message A with $c \neq \tilde{c}$ and

$$A = ((y_1, y_2), (y_{1j}, y_{2j}, y_{3j}, y_{4j})_{j=0}^{2k-1}, ((y_{1ij}, y_{2ij})_{j=0}^{2k-1})_{i=0,1,2})$$

and

$$B = ((s, \rho_1', \rho_2'), ((c_{\lambda j}, \rho_{\lambda_j})_{\lambda=1,2,3,4})_{j=0}^{2k-1}, ((c_{1ij}, c_{2ij}, \rho_{1ij}, \rho_{2ij})_{j=0}^{2k-1})_{i=0,1,2})$$

and

$$\tilde{B} = ((\tilde{s}, \tilde{\rho_1'}, \tilde{\rho_2'}), ((\tilde{c_{\lambda j}}, \tilde{\rho_{\lambda_j}})_{\lambda=1,2,3,4})_{j=0}^{2k-1}, ((\tilde{c_{1ij}}, \tilde{c_{2ij}}, \tilde{\rho_{1ij}}, \tilde{\rho_{2ij}})_{j=0}^{2k-1})_{i=0,1,2}).$$

Then for condition (ai) of step 4, since $c \neq \tilde{c}$, for $j = 0, \dots, 2k-1$ and $i = 0,1,2$ either $c_{1ij} \neq \tilde{c_{1ij}}$ or $c_{2ij} \neq \tilde{c_{2ij}}$. In the first case we have $h^{\rho_{1ij}} = y_{1ij}(C_{i_j})^{c_{1ij}}$ and $h^{\tilde{\rho_{1ij}}} = y_{1ij}(C_{i_j})^{\tilde{c_{1ij}}}$ whose division gives us

$$h^{\rho_{1ij}-\tilde{\rho_{1ij}}} = (C_{i_j})^{c_{1ij}-\tilde{c_{1ij}}} \Leftrightarrow h^{\frac{\rho_{1ij}-\tilde{\rho_{1ij}}}{c_{1ij}-\tilde{c_{1ij}}}} = C_{i_j}$$

i.e. we extract $\zeta = \frac{\rho_{1ij}-\tilde{\rho_{1ij}}}{c_{1ij}-\tilde{c_{1ij}}}$ as a valid opening for C_{i_j}. In the second case we have that $h^{\rho_{2ij}} = y_{2ij}(C_{i_j}/g)^{c_{2ij}}$ and $h^{\tilde{\rho_{2ij}}} = y_{2ij}(C_{i_j}/g)^{\tilde{c_{2ij}}}$ and similarly $h^{\frac{\rho_{2ij}-\tilde{\rho_{2ij}}}{c_{2ij}-\tilde{c_{2ij}}}} = (C_{i_j}/g)$, i.e.we extract $\zeta' = \frac{\rho_{2ij}-\tilde{\rho_{2ij}}}{c_{2ij}-\tilde{c_{2ij}}}$ as a valid opening for (C_{i_j}/g).

From condition (aii) for $j = 0, \dots, 2k-1$ and $\lambda = 1,2,3,4$ we have that $h^{\rho_{\lambda j}} = y_{\lambda j}(\frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda})^{c_{\lambda j}}$ and $h^{\tilde{\rho_{\lambda j}}} = y_{\lambda j}(\frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda})^{\tilde{c_{\lambda j}}}$ whose division yields $h^{\rho_{\lambda j} - \tilde{\rho_{\lambda j}}} = (\frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda})^{c_{\lambda j} - \tilde{c_{\lambda j}}}$. Since $c \neq \tilde{c}$ for each $j = 0, \dots, 2k-1$ there is a $\lambda \in \{1,2,3,4\}$ such that $c_{\lambda j} \neq \tilde{c_{\lambda j}}$. Thus $h^{\frac{\rho_{\lambda j} - \tilde{\rho_{\lambda j}}}{c_{\lambda j} - \tilde{c_{\lambda j}}}} = (\frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda})$, concluding that we can extract $\eta = \frac{\rho_{\lambda j} - \tilde{\rho_{\lambda j}}}{c_{\lambda j} - \tilde{c_{\lambda j}}}$ as a valid opening for $C'_j = \frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda}$.

Finally from condition (b) we have that $g^s pk_t^{\rho'_1} = y_1(E_x)^c$ and $g^{\tilde{s}} pk_t^{\tilde{\rho'_1}} = y_1(E_x)^{\tilde{c}}$ from which we have that $g^{\frac{s-\tilde{s}}{c-\tilde{c}}} h^{\frac{\rho'_1 - \tilde{\rho'_1}}{c-\tilde{c}}} = E_x$, i.e. we extract $\alpha = \frac{s-\tilde{s}}{c-\tilde{c}}$, $\beta = \frac{\rho'_1 - \tilde{\rho'_1}}{c-\tilde{c}}$ as a valid opening for $E_x = g^\alpha h^\beta$.

Similarly from

$$g^s h^{\rho'_2} = y_2(\prod_{j=0}^{k-1}(\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{(2^j)})^c \text{ and } g^{\tilde{s}} h^{\tilde{\rho'_2}} = y_2(\prod_{j=0}^{k-1}(\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{(2^j)})^{\tilde{c}}$$

we extract $\alpha = \frac{s-\tilde{s}}{c-\tilde{c}}$ and $\beta' = \frac{\rho'_2 - \tilde{\rho'_2}}{c-\tilde{c}}$ as a valid opening for

$$C'' = (\prod_{j=0}^{k-1}(\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{2^j}).$$

HV Zero Knowledge: There is a simulator that can simulate the communication transcript. On input G, q, g, pk_t, h, E_x, $\{(C_{i_j})_{j=0,\dots,2k-1}\}_{i=0,1,2}$ it picks randomly a value $c \in \mathbb{Z}_q$. It also picks uniformly at random $s, \rho'_1, \rho'_2 \in \mathbb{Z}_q$. For $\lambda = 1,2,3,4$ and for $j = 0, \dots, 2k-1$ it picks $c_{\lambda j}, \rho_{\lambda j} \in Z_q$ such that $\sum_{\lambda=1}^4 c_{\lambda j} = c$. For $i = 0,1,2$ and $j = 0, \dots, 2k-1$ it picks $c_{1ij}, c_{2ij}, \rho_{1ij}, \rho_{2ij} \in \mathbb{Z}_q$ such that $c_{1ij} + c_{2ij} = c$. We define the second and the third message of the communication transcript to be respectively c and

$$B = ((s, \rho'_1, \rho'_2), ((c_{\lambda j}, \rho_{\lambda_j})_{\lambda=1,2,3,4})_{j=0}^{2k-1}, ((c_{1ij}, c_{2ij}, \rho_{1ij}, \rho_{2ij})_{j=0}^{2k-1})_{i=0,1,2}).$$

We now set the first message of the transcript to be A:

- $(y_1, y_2) = (g^s pk_t^{\rho'_1}(E_x)^{-c}, g^s h^{\rho'_2}(\prod_{j=0}^{k-1}(\frac{g^2}{C_{0_{2j+1}} \cdot C_{0_{2j}}})^{(2^j)})^{-c})$
- for $j = 0, \dots, 2k-1$, for $\lambda = 1,2,3,4$: $y_{\lambda j} = h^{\rho_{\lambda j}}(\frac{(C_{1_j})(C_{2_j})^2(C_{0_j})^3}{d_\lambda})^{-c_{\lambda j}}$
- For $i = 0,1,2$, for $j = 0, \dots, 2k-1$:

$$(y_{1ij}, y_{2ij}) = (h^{\rho_{1ij}}(C_{i_j})^{-c_{1ij}}, h^{\rho_{2ij}}(C_{i_j}/g)^{-c_{2ij}}).$$

The final simulated transcript is $\langle A, c, B \rangle$.