Making Sigma-protocols Non-interactive without Random Oracles*

Pyrros Chaidos** and Jens Groth***

University College London

Abstract. Damgård, Fazio and Nicolosi (TCC 2006) gave a transformation of Sigma-protocols, 3-move honest verifier zero-knowledge proofs, into efficient non-interactive zero-knowledge arguments for a designated verifier. Their transformation uses additively homomorphic encryption to encrypt the verifier's challenge, which the prover uses to compute an encrypted answer. The transformation does not rely on the random oracle model but proving soundness requires a complexity leveraging assumption.

We propose an alternative instantiation of their transformation and show that it achieves culpable soundness without complexity leveraging. This improves upon an earlier result by Ventre and Visconti (Africacrypt 2009), who used a different construction which achieved *weak* culpable soundness.

We demonstrate how our construction can be used to prove validity of encrypted votes in a referendum. This yields a voting system with homomorphic tallying that does not rely on the Fiat-Shamir heuristic.

Keywords: Sigma-protocols, non-interactive zero-knowledge designated verifier argument, DFN transformation, culpable soundness, voting.

1 Introduction

Cryptographic applications often require a party to demonstrate that a statement is true without revealing any additional details. For example, a voter may wish to prove that an encrypted message contains a vote for a valid candidate without disclosing the actual candidate. This can be done using *zero-knowledge* proofs [17] that enable a prover to demonstrate to a verifier that a statement x belongs to a language L in NP defined by a relation R without giving the verifier any information about the witness w such that $(x, w) \in R$.

* ©IACR 2015. This article is the final version submitted by the authors to the IACR and to Springer-Verlag on January 11, 2015. The version published by Springer-Verlag is available at http://dx.doi.org/10.1007/978-3-662-46447-2_29

** This author was supported by an EPSRC scholarship (EP/G037264/1 – Security Science DTC)

*** This research was supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937.

Σ-protocols are particular types of 3-move honest verifier zero-knowledge proofs that can be highly efficient. However, in many applications it is preferable for a protocol to be non-interactive [5] with the prover preparing a proof with no need for direct input from the verifier. The Fiat-Shamir transformation [13] produces a non-interactive version of a Σ-protocol by substituting the verifier's challenge with the output of a hash function on the prover's statement and messages. The transformation can be proven secure in the random oracle model [3]. However, the random oracle model is regarded with some skepticism since there exist pathological protocols that can be proven secure in the random oracle model but fail in any real-world instantiation [6, 16].

Damgård, Fazio and Nicolosi (DFN) [11] introduced an alternative transformation. The DFN transformation works in the Registered Key Model (RKM) [2] where a verifier registers a public key and transforms a Σ-protocol with linear answer into a non-interactive zero-knowledge argument that can be verified by this specific verifier [23]. The transformation works by having the verifier encrypt his challenge under an additively homomorphic encryption scheme and relies on the Σ-protocol having an answer that can be computed using linear algebra and the homomorphic property of the encryption scheme to enable the prover to complete an encrypted version of the answer in the Σ-protocol. Their construction is secure for a logarithmic number of proofs but soundness rests on a complexity leveraging assumption.

Ventre and Visconti [28] give an alternative proof of soundness for a construction based on a two ciphertext variation of the DFN transformation in the style of Naor and Yung [25]. They replace the complexity leveraging assumption by introducing a modification of culpable soundness[1] [21] that they call *weak culpable soundness*. Standard culpable soundness restricts adversaries to being "aware" of the falsehood of the statement they are proving. Weak culpable soundness furthermore requires that the adversary is also aware of the fact that she has succeeded in producing a convincing proof of a false statement, by producing a second auxiliary proof to that effect.

In the DFN setting using weak culpable soundness would require the adversary to prove statements containing ciphertexts addressed to the designated verifier. It would be challenging to provide such an adversary with enough power to perform the required proofs without having knowledge of the verifier's secret decryption key. We instead opt to construct the underlying protocol with the property that forged proofs reveal the challenge. This is enough to contradict the semantic security of the encryption scheme used for the designated verifier proof if a false proof is ever produced.

1.1 Our Contribution

We give an instantiation of the DFN transformation that achieves standard culpable soundness without complexity leveraging. The transformation relies on an IND-CPA secure additively homomorphic encryption scheme and is quite

[1] Culpable soundness was also called co-soundness in an earlier version of [21].

efficient. The tranformation can be applied to Σ-protocols that have linear answers and unique identifiable challenges (Sect. 2.2).

We can use our resulting non-interactive zero-knowledge designated verifier arguments to efficiently prove statements about encrypted plaintexts. In particular, we can prove that a ciphertext contains either 0 or 1 without disclosing the plaintext. This can in turn be used to prove that a set of ciphertexts encrypt a witness for the satisfiability of a circuit. For the appropriate Σ-protocols to be in place, we require the encryption scheme to be additively homomorphic modulo a prime and satisfy a few other requirements (Sect. 2.1). We use Okamoto-Uchiyama encryption [26] as an example.

We proceed to give an example application of our non-interactive zero-knowledge arguments to provide publicly verifiable arguments in the context of electronic voting. In voting systems such as Helios [1] voters submit their votes encrypted under a homomorphic encryption scheme accompanied with non-interactive arguments (typically using the Fiat-Shamir transformation) that the encrypted votes are in fact valid. Ciphertexts with convincing arguments are aggregated homomorphically to produce an encrypted tally which is then decrypted to produce the result. By releasing the designated verifier keys to the public (similar to [29]), once vote submission has concluded, we can use our non-interactive designated verifier arguments in place of the usual non-interactive zero-knowledge arguments with minimal changes to the design.

1.2 Related Work

Since the introduction of non-interactive zero-knowledge proofs by Blum, Feldman and Micali [5] much effort has been spent on reducing their size [10, 24, 19, 15]. The introduction of pairing-based techniques [21, 18, 22] has led to practically efficient non-interactive zero-knowledge proofs that can be used in the context of pairing-based cryptography.

The Fiat-Shamir heuristic can be used to make a Σ-protocol non-interactive. This can lead to highly efficient non-interactive zero-knowledge proofs but relies on the random oracle model when proving security. Recently pairing-based succinct non-interactive zero-knowledge arguments [20, 14, 27] have become very compact even for large scale statements, however, they rely on knowledge extractor assumptions over bilinear groups.

The above research yields non-interactive zero-knowledge proofs that are publicly verifiable. However, there are many settings where it suffices to have non-interactive zero-knowledge arguments intended for a designated verifier. Cramer and Shoup used universal hash proofs to build a highly efficient chosen ciphertext attack secure public-key encryption scheme [8, 9]. Non-interactive proofs for a designated verifier for all languages in NP can be found in [2] in the key registration model where parties register keys.

The most closely related works are the DFN transformation by Damgård, Fazio and Nicolosi [11] and the work by Ventre and Visconti [28] that we have already discussed.

2 Preliminaries

We write $y = A(x; r)$ when the algorithm A on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S.

All algorithms get as input a security parameter n written in unary as 1^n. Sometimes we do not explicitly write this input to the algorithms but we will always assume it is implicitly available to the algorithms. The intuition is that the higher the security parameter, the more secure the cryptographic system.

Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(n) \approx g(n)$ when $|f(n)-g(n)| = O(n^{-c})$ for every constant $c > 0$. We say that f is *negligible* if $f(n) \approx 0$ and that f is *overwhelming* if $f(n) \approx 1$.

An NP-relation is a binary relation R consisting of pairs (x, w) that can be decided in polynomial time in the length of x. We call x the statement and w the witness. The relation R gives rise to a language $L_R = \{x \mid \exists w : (x, w) \in R\}$ of statements in R. To incorporate the security parameter into the relations, we will without loss of generality assume all statements are of a form such that n can be easily derived (all statements in this paper could be reformulated to be of the form $x = (1^n, x')$ although for notational convenience we will not do this) and all statements and witnesses are of size polynomial in n. We define R_n as the relation R restricted to statements corresponding to n.

2.1 Additively Homomorphic Encryption

A public key encryption scheme is a triple of probabilistic polynomial time algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation function $\mathcal{K}$ given a security parameter returns a public encryption key ek and a private decryption key dk. The encryption algorithm $\mathcal{E}$ given an encryption key ek and a message m returns a ciphertext $c \leftarrow \mathcal{E}_{ek}(m)$. The deterministic decryption algorithm $\mathcal{D}$ given a decryption key dk and a ciphertext c returns a message m or a special symbol $\perp$ if the ciphertext is invalid.

The public encryption key ek defines a message space $\mathcal{M}_{ek}$ of possible plaintexts, a randomness space $\mathcal{R}_{ek}$ and a ciphertext space $\mathcal{C}_{ek}$. In this paper we will make use of an encryption scheme where the message space is $\mathbb{Z}_p$ for some large integer p, which is explicitly or implicitly defined by the public key, and with size $|p| = \ell_p(n)$ for a publicly known polynomial ℓ_p. We say that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is *additively homomorphic* if the randomness and ciphertext spaces are finite groups as well (written additively and multiplicatively respectively) and for all possible keys ek and plaintexts $m_1, m_2 \in \mathcal{M}_{ek}$ and $r_1, r_2 \in \mathcal{R}_{ek}$ we have

$$\mathcal{E}_{ek}(m_1; r_1) \cdot \mathcal{E}_{ek}(m_2; r_2) = \mathcal{E}_{ek}(m_1 + m_2; r_1 + r_2).$$

We say that an additively homomorphic scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a *strongly additively homomorphic scheme* if it satisfies the four additional properties described below:

Prime order message space: The message space is $\mathbb{Z}_p$ for some *prime* p.

Decryption homomorphic[2]**:** Membership of the ciphertext space can be efficiently tested and the decryption algorithm on all elements in $\mathcal{C}_{ek}$ returns a plaintext in $\mathcal{M}_{ek}$ (i.e., decryption does not fail). Furthermore, decryptions respect the additively homomorphic operation, i.e., for all possible key pairs (ek, dk) and $c_1, c_2 \in \mathcal{C}_{ek}$ we have

$$\mathcal{D}_{dk}(c_1) + \mathcal{D}_{dk}(c_2) = \mathcal{D}_{dk}(c_1 \cdot c_2).$$

Extended randomness: $R_{ek} = \mathbb{Z}_N$ for some integer N but the encryption function accepts randomness in $\mathbb{Z}$ and for all $m \in \mathcal{M}_{ek}$ and $r \in \mathbb{Z}$

$$\mathcal{E}(m; r) = \mathcal{E}(m; r \bmod N).$$

Verifiable keys: There exists an efficient test VerifyKey$(1^n, ek, dk)$ that given a public key ek and decryption key dk (or without loss of generality the randomness used in the key generation) returns 1 if and only if (ek, dk) is a valid key pair using security parameter n.

For notational convenience, we let $c^{\boldsymbol{z}}$ be the vector $(c^{z_1}, \ldots, c^{z_n})$ given a ciphertext c and a vector of integers $\boldsymbol{z} = (z_1, \ldots, z_n)$. Given a vector $\boldsymbol{w}$ we also define $\boldsymbol{c} \leftarrow \mathcal{E}_{ek}(\boldsymbol{w})$ as the vector of ciphertexts given by $(\mathcal{E}_{ek}(w_1), \ldots, \mathcal{E}_{ek}(w_n))$.

Definition 1 (IND-CPA security). *We say that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is indistinguishable under chosen plaintext attack (IND-CPA secure) if for all probabilistic polynomial time stateful adversaries $\mathcal{A}$*

$$\Pr\left[(ek, dk) \leftarrow \mathcal{K}(1^n); (m_0, m_1) \leftarrow \mathcal{A}(ek); b \leftarrow \{0,1\}; c \leftarrow \mathcal{E}_{ek}(m_b) : \mathcal{A}(c) = b\right] \approx \frac{1}{2},$$

where $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}_{ek}$.

Okamoto-Uchiyama encryption [26] The Okamoto-Uchiyama [26] cryptosystem is strongly additively homomorphic with a message space $\mathbb{Z}_p$ for a prime p that is implicitly defined by the public key.

$\mathcal{K}(1^n)$**:** Pick two different $\ell_p(n)$-bit primes p, q and let $N = p^2 \cdot q$. Then choose a random g in $\mathbb{Z}_N^*$ such that $g \bmod p^2$ has order $p(p-1)$ in $\mathbb{Z}_{p^2}^*$. The public key is $ek = (N, g)$ and the secret decryption key is $dk = (ek, p)$.

$\mathcal{E}_{ek}(m)$**:** Given $m \in \mathbb{Z}_p$ return $\mathcal{E}_{ek}(m; r) = g^{m+rN} \bmod N$, where $r \leftarrow \mathbb{Z}_N$.

$\mathcal{D}_{dk}(c)$**:** Return $m = \frac{L(c^{p-1} \bmod p^2)}{L(g^{p-1} \bmod p^2)} \bmod p$, where $L(x) = \frac{x-1}{p}$.

For a given public key $ek = (N, g)$ the randomness space is $\mathbb{Z}_N$ and the ciphertext space is $\mathbb{Z}_N^*$. Even though the message space is defined as $\mathbb{Z}_p$, in

[2] This property is trivial for cryptosystems where the entire cipherspace consists of valid encryptions but in the general case it must be stated explicitly.

practice we cannot disclose p but as long as the encrypting party picks messages $m \in \{0,1\}^{\ell_p(n)-1}$ we are guaranteed that they fall within the message space and will decrypt correctly.

Direct calculation confirms that Okamoto-Uchiyama encryption is decryption homomorphic and that it is easy to extend the randomness space to $R_{ek} = \mathbb{Z}$. The keys are verifiable in the sense that given the decryption key, i.e., the factorization of N, it is easy to check that the keys are a valid output of the key generation algorithm and that the encryption scheme satisfies all the required properties.

2.2 Σ-protocols with Linear Answers and Unique Identifiable Challenges

A Σ-protocol for an NP-relation R is a 3-move protocol that enables a prover to demonstrate to a verifier that a statement x satisfies $x \in L_R$, i.e. that there exists w such that $(x, w) \in R$ without disclosing anything else, in particular not disclosing the value of w that the prover has in mind. A typical run of a Σ-proocol is illustrated in Fig. 1.

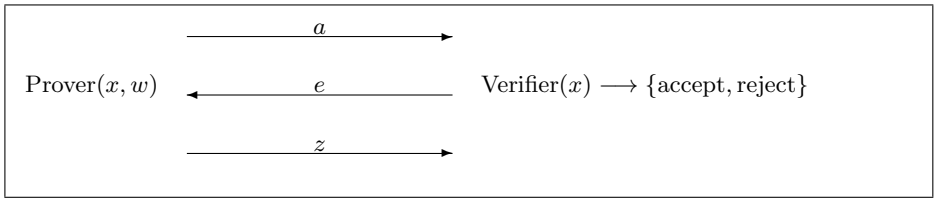

Fig. 1. Σ-protocol with statement x and witness w.

A Σ-protocol is *public-coin*, which means that the challenge e chosen by the verifier is picked uniformly at random without the verifier storing any private information about it. We will consider protocols where e is picked as a random n-bit string, where n is the security parameter.

We will restrict ourselves to Σ-protocols with a *linear answer over the integers*. By this we mean without loss of generality that we can consider a prover that generates the initial message a and two integer vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The answer to a challenge $e \in \{0,1\}^n$ can then be computed as the integer vector $\boldsymbol{z} = e\boldsymbol{z}_1 + \boldsymbol{z}_2$. We will assume that all the integers in $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}$ are non-negative and that there is a known polynomial upper bound $\ell_z(n)$ on the bit-size of the integers.

We can now describe a Σ-protocol for an NP-relation R with linear answer as a pair $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$, where $\mathcal{P}_\Sigma, \mathcal{V}_\Sigma$ are probabilistic polynomial time algorithms. The Σ-protocol runs as follows:

$(a, \boldsymbol{z}_1, \boldsymbol{z}_2) \leftarrow \mathcal{P}_\Sigma(x, w)$**:** The prover given a statement and witness pair $(x, w) \in R_n$ generates an initial message a and a state $\boldsymbol{z}_1, \boldsymbol{z}_2$.

$e \leftarrow \{0,1\}^n$**:** An n-bit challenge is chosen uniformly at random.

$\boldsymbol{z} \leftarrow e\boldsymbol{z}_1 + \boldsymbol{z}_2$**:** An answer to the challenge e can be computed as $\boldsymbol{z} = e\boldsymbol{z}_1 + \boldsymbol{z}_2$.

$\{0,1\} \leftarrow \mathcal{V}_\Sigma(x, a, e, \boldsymbol{z})$**:** The verifier given a statement x and a protocol transcript $(a, e, \boldsymbol{z})$ returns 1 if accepting and 0 if rejecting. The verifier will always reject if any inputs are malformed, for instance if $e \notin \{0,1\}^n$ or $\boldsymbol{z}$ contains an entry $z_i \notin \{0,1\}^{\ell_z(n)}$.

A Σ-protocol is required to operate correctly when used by honest participants (*completeness*), to prevent dishonest provers from convincing verifiers that false statements hold (*soundness*), and not to leak information about w (*zero-knowledge*). Formally, we require that a Σ-protocol $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ for an NP-relation R with linear answer should be complete and special honest verifier zero-knowledge as defined below. With respect to soundness, we will for our purposes be interested in a special class of Σ-protocols that have unique identifiable challenges.

Definition 2 (Completeness). *We say $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ is perfectly complete if for all $n \in \mathbb{N}$ and $(x, w) \in R_n$*

$$\Pr\left[(a, \boldsymbol{z}_1, \boldsymbol{z}_2) \leftarrow \mathcal{P}_\Sigma(x, w); e \leftarrow \{0,1\}^n; \boldsymbol{z} = e\boldsymbol{z}_1 + \boldsymbol{z}_2 : \mathcal{V}_\Sigma(x, a, e, \boldsymbol{z}) = 1\right] = 1.$$

Definition 3 (Special Honest Verifier Zero-Knowledge (SHVZK)). *We say that $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ is computationally special honest verifier zero-knowledge if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all probabilistic polynomial time stateful adversaries $\mathcal{A}$*

$$\begin{aligned}&\Pr\left[\begin{array}{c}(x, w) \leftarrow \mathcal{A}(1^n); (a, \boldsymbol{z}_1, \boldsymbol{z}_2) \leftarrow \mathcal{P}_\Sigma(x, w); e \leftarrow \{0,1\}^n; \boldsymbol{z} \leftarrow e\boldsymbol{z}_1 + \boldsymbol{z}_2 : \\ (x, w) \in R_n \text{ and } \mathcal{A}(a, e, \boldsymbol{z}) = 1\end{array}\right] \\ \approx &\Pr\left[\begin{array}{c}(x, w) \leftarrow \mathcal{A}(1^n); e \leftarrow \{0,1\}^n; (a, \boldsymbol{z}) \leftarrow \mathcal{S}(x, e) : \\ (x, w) \in R_n \text{ and } \mathcal{A}(a, e, \boldsymbol{z}) = 1\end{array}\right]\end{aligned}$$

If this holds also for unbounded adversaries $\mathcal{A}$, we say $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ is statistically special honest verifier zero-knowledge.

Traditionally, Σ-protocols are required to have *special soundness*, which says that if the prover, after having created the initial message a, can answer two different challenges e and e' then it is possible to compute a witness w for the statement x being proved such that $(x, w) \in R$.

We do not need the witness to be extractable in this paper and will therefore relax the soundness definition to just saying that on a false statement there is at most a single unique challenge the prover can answer after having created the initial message a.

However, we will require that under certain circumstances this unique answerable challenge should be identifiable, i.e., if the prover "knows" the statement is false in a certain way then she can actually compute the unique challenge e she will be able to answer if she can answer any challenge at all. We define this by adapting the notion of culpable soundness from [21]. We say that the unique challenge is identifiable using an NP-relation R_{guilt}, which only contains

false statements, if when the prover produces a statement x and a witness w_{guilt} of being guilty of cheating such that $(x, w_{\text{guilt}}) \in R_{\text{guilt}}$, then it is possible to efficiently compute a unique challenge where the verifier may possibly accept. The relation R_{guilt} will typically include all false statements that have a special form, depending on the specifics.

Definition 4 (Soundness with unique identifiable challenge). *We say $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ has a unique identifiable challenge using NP-relation R_{guilt} if there is a polynomial time algorithm E that takes as input the statement, witness and initial message and returns the unique challenge e that can be answered. Formally, we require that for all $n, x, w_{\text{guilt}}, a, e, z$ where $(x, w_{\text{guilt}}) \in R_{\text{guilt},n}$ and $\mathcal{V}_\Sigma(x, a, e, z) = 1$ that $e = E(x, w_{\text{guilt}}, a)$.*

A frequently asked question is why would the adversary want to provide a witness for cheating. The answer is that there are many natural scenarios where the real adversary is only a part of a larger system that contains the guilt witness. It may well be that the system would never provide a guilt witness in a normal execution but even when that is the case the notion can still be useful in security proofs: by framing a "standard" adversary within such a system we are able to explicitly use privileged information held by honest parties in security reductions. In Sect. 4 we give voting as a concrete example of how culpable soundness can be used to prevent cheating by voters. Voters prove that they have encrypted valid votes using the election system's public key. The guilt witness is the decryption key, which the voting system will never make public since it would reveal all the votes. However, if a cheating voter exists, it is enough to point out that the guilt witness will exist in the possession of the electoral authorities. To satisfy the definition we may consider a new adversary which consists of the cheating voter's behaviour, with the decryption key added to the output in a post-processing step. Culpable soundness then guarantees the voter cannot cheat and submit an invalid vote.

We note that the extractor E only requires the guilt witness and the initial message from the prover. This will be critical in the next section where the protocol is made non-interactive via the DFN transformation and the prover's answer will be encrypted. In general, we cannot require that a cheating prover knows the contents of that ciphertext since it might have been assembled in a way that differs from the protocol.

Σ-protocol for additively homomorphic encryption of 0 or 1. Consider a strongly additively homomorphic encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ with message space $\mathbb{Z}_p$ for a prime p defined by the encryption key. We will now give a Σ-protocol for proving that a ciphertext encrypts 0 or 1 using randomness $r \in \{0,1\}^{\ell_r(n)}$ bounded by a polynomial $\ell_r(n)$.

Let

$$R = \left\{ \Big((ek, c), (m, r) \Big) : m \in \{0,1\} \text{ and } r \in \{0,1\}^{\ell_r(n)} \text{ and } c = \mathcal{E}_{ek}(m; r) \right\},$$

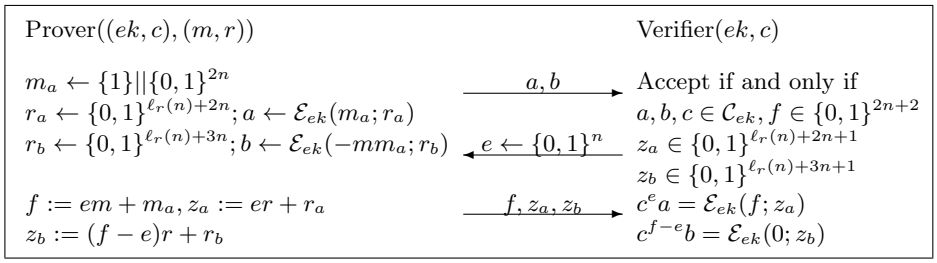

Fig. 2. Σ-protocol for encryption of 0 or 1.

$$R_{\text{guilt}} = \left\{ \Big((ek, c), dk \Big) : c \in \mathcal{C}_{ek} \text{ and } \mathcal{D}_{dk}(c) \notin \{0,1\} \text{ and } \text{VerifyKey}(1^n, ek, dk) = 1 \right\}.$$

Theorem 1. *Fig. 2 describes a Σ-protocol for R with linear answer and unique identifiable challenge using R_{guilt} assuming $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a strongly additively homomorphic encryption scheme with message space $\mathbb{Z}_p$ of sufficiently large size such that $\ell_p(n) > n$.*

Proof. The algorithms are probabilistic polynomial time. The protocol has linear answer with a polynomial upper bound of $\ell_z(n) = \ell_r(n) + 3n + 1$ on the bit-lengths of the integers in the answer. Direct verification shows that the protocol is perfectly complete.

The protocol is statistical SHVZK. The simulator given challenge $e \in \{0,1\}^n$ picks $f \leftarrow \{1\}||\{0,1\}^{2n}$, $z_a \leftarrow \{0,1\}^{\ell_r(n)+2n}$ and $z_b \leftarrow \{0,1\}^{\ell_r(n)+3n}$. It then computes $a = c^{-e}\mathcal{E}_{ek}(f; z_a)$ and $b = c^{e-f}\mathcal{E}_{ek}(0; z_b)$ and returns the simulated proof (a, b, f, z_a, z_b). Observe that the simulated f, z_a, z_b are statistically close to those of a real proof. To see the simulation is statistically indistinguishable from a real proof with challenge e all that remains to be seen is that given f, z_a, z_b, the initial message containing a, b is fixed by the verification equations in both real and simulated proofs.

Finally, let us show that the protocol has unique identifiable challenges using R_{guilt}. A witness in R_{guilt} gives us the decryption key for the encryption scheme. We can verify the correctness of the decryption key and decrypt c to get m and also decrypt a, b to get plaintexts m_a and m_b. In a succesful argument, the value f must be $f = em + m_a \bmod p$ since otherwise the first verification equation would fail. The second verification equation gives us $(f - e)m + m_b = 0 \bmod p$, which means $e(m-1)m + m_a m + m_b = 0 \bmod p$. If $m \notin \{0,1\}$ we have that $(m-1)m \neq 0 \bmod p$ and therefore the equation uniquely determines $e \bmod p$. With $p > 2^n$ this identifies at a unique challenge $e \in \{0,1\}^n$ that the prover may be able to answer or shows that no answerable challenge exists. □

2.3 Non-Interactive Designated Verifier Zero-Knowledge Arguments

It is often desirable to operate in a single step, avoiding the interaction needed to execute a Σ-protocol. The prover still wishes to demonstrate to the verifier

the truth of a statement $x \in L_R$ for an NP-relation R without disclosing any other information about her witness w.

In a non-interactive designated verifier zero-knowledge argument system, we imagine the verifier sets up a public key pk for the proof together with a secret verification key vk that can be used to verify the arguments. The system therefore consists of three probabilistic polynomial time algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$.

$(pk, vk) \leftarrow \mathcal{G}(1^n)$**:** The key generation algorithm, given the security parameter as input, generates a public key pk and a secret verification key vk.

$\pi \leftarrow \mathcal{P}(pk, x, w)$**:** Given a public key pk and $(x, w) \in R_n$, the prover algorithm generates an argument π.

$\{0, 1\} \leftarrow \mathcal{V}(vk, x, \pi)$**:** Given a secret verification key vk, a statement x and an argument π, the verification algorithm returns 1 if accepting the argument and 0 for rejection of the argument.

$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is said to be a non-interactive designated verifier zero-knowledge argument system for R with culpable soundness with respect to R_{guilt} if it is complete, culpably sound and zero-knowledge as defined below.

Definition 5 (Completeness). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is perfectly complete if for all* $n \in \mathbb{N}$ *and all* $(x, w) \in R_n$

$$\Pr\left[(pk, vk) \leftarrow \mathcal{G}(1^n); \pi \leftarrow \mathcal{P}(pk, x, w) : \mathcal{V}(vk, x, \pi) = 1\right] = 1.$$

Intuitively, the argument is zero-knowledge if it does not leak information about the witness. The arguments we construct will be zero-knowledge assuming the keys are honestly generated. We define this notion through the existence of a simulator that can simulate arguments given the verifier's secret verification key. In our constructions we will get zero-knowledge even if the adversary knows the secret verification key, a strong type of zero-knowledge called composable zero-knowledge in [18] due to it making composition of zero-knowledge proofs easier.

Definition 6 (Composable zero-knowledge). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is computationally composable zero-knowledge if for all probabilistic polynomial time stateful adversaries* $\mathcal{A}$

$$\begin{aligned}&\Pr\left[(pk, vk) \leftarrow \mathcal{G}(1^n); (x, w) \leftarrow \mathcal{A}(pk, vk); \pi \leftarrow \mathcal{P}(pk, x, w) : (x, w) \in R_n \text{ and } \mathcal{A}(\pi) = 1\right] \\ \approx &\Pr\left[(pk, vk) \leftarrow \mathcal{G}(1^n); (x, w) \leftarrow \mathcal{A}(pk, vk); \pi \leftarrow \mathcal{S}(vk, x) : (x, w) \in R_n \text{ and } \mathcal{A}(\pi) = 1\right].\end{aligned}$$

If the above holds also for unbounded stateful adversaries $\mathcal{A}$ *then we say the argument is statistically composable zero-knowledge.*

Culpable soundness [21] is a relaxation of soundness that restricts the prover in the following way: First, we only consider false statements in a subset L_{guilt} of $\bar{L}_R$ characterised by a relation R_{guilt}. Second, we require a successful cheating

prover to also output a guilt witness w_{guilt} along with his false statement x such that $(x, w_{\text{guilt}}) \in R_{\text{guilt}}$. Intuitively this definition captures the notion of a malicious prover being aware of the falsehood of the statement for which she is creating a fake proof.

Definition 7 (Adaptive culpable soundness). *We say $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is culpably sound with respect to the relation R_{guilt} if for all probabilistic polynomial time $\mathcal{A}$*

$$\Pr\left[(pk, vk) \leftarrow \mathcal{G}(1^n); (x, \pi, w_{\text{guilt}}) \leftarrow \mathcal{A}(pk) : (x, w_{\text{guilt}}) \in R_{\text{guilt},n} \text{ and } \mathcal{V}(vk, x, \pi) = 1\right] \approx 0.$$

The above definition does not directly cover the adversary, $\mathcal{A}$ having access to a verification oracle $\mathcal{V}(vk, \cdot, \cdot)$. However it is straightforward to handle cases where the adversary has access to a logarithmic number of queries (as in [11]), since that can be simulated by guessing the responses with inverse polynomial probability.

3 Transformation

We will now use the DFN transformation on a Σ-protocol with linear answer over the integers and unique identifiable challenges to get a non-interactive designated verifier argument. The verifier uses an additively homomorphic encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ to encrypt a random challenge e. Since the Σ-protocol has linear answer, the prover can now use the homomorphic property of the encryption scheme to compute an encryption of the answer $\boldsymbol{z}$ in the Σ-protocol, which is sent together with the initial message a. The verifier decrypts the ciphertext from the prover to get z and checks whether $(a, e, \boldsymbol{z})$ is a valid proof. The full non-interactive designated verifier argument is described in Fig. 3.

$\mathcal{G}(1^n)$	$\mathcal{P}(pk, x, w)$	$\mathcal{V}(vk, x, \pi)$
$(ek, dk) \leftarrow \mathcal{K}(1^n)$	$(a, \boldsymbol{z}_1, \boldsymbol{z}_2) \leftarrow \mathcal{P}_\Sigma(x, w)$	Parse $\pi = (a, \boldsymbol{c}_z)$
$e \leftarrow \{0,1\}^n$	$\boldsymbol{c}_z \leftarrow c^{\boldsymbol{z}_1} \mathcal{E}_{ek}(\boldsymbol{z}_2)$	$\boldsymbol{z} \leftarrow \mathcal{D}_{dk}(\boldsymbol{c}_z)$
$c \leftarrow \mathcal{E}_{ek}(e)$	Return $\pi := (a, \boldsymbol{c}_z)$	Return $\mathcal{V}_\Sigma(x, a, e, \boldsymbol{z})$
$pk := (ek, c)$		
$vk := (dk, e)$		
Return (pk, vk)		

Fig. 3. Non-interactive designated verifier argument

Theorem 2. *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ specified in Fig. 3 is a non-interactive designated verifier argument for R with culpable soundness for R_{guilt} if $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ is a Σ-protocol for R with linear answer over the integers and soundness with unique identifiable challenge using R_{guilt} and if $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an additively homomorphic, IND-CPA secure public key encryption scheme where $\mathbb{Z}_p$ is of sufficiently large size to include the answers, i.e., $\ell_p(n) > \ell_z(n)$.*

Proof. Since $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$ and $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ are probabilistic polynomial time algorithms so are $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. Perfect completeness follows from the additive homomorphicity of the encryption scheme and that $0 \leq z_i < 2^{\ell_z(n)} < p$ for all entries z_i in $\boldsymbol{z}$ combined with the perfect completeness of $(\mathcal{P}_\Sigma, \mathcal{V}_\Sigma)$.

Next, we will prove that the construction is zero-knowledge. The simulator knows the secret verification key $vk = (dk, e)$. It starts by running the SHVZK simulator for the Σ-protocol to get a simulated proof $(a, e, \boldsymbol{z})$ for the statement x. It then generates $\boldsymbol{c}_z \leftarrow \mathcal{E}_{ek}(\boldsymbol{z})$ and returns the simulated argument $\pi := (a, \boldsymbol{c}_z)$.

To see a that simulated argument is indistinguishable from a real argument consider a hybrid simulator that does get the witness as input. This hybrid simulator proceeds by following the Σ-protocol to get an argument $(a, e, \boldsymbol{z})$ and then encrypts $\boldsymbol{z}$ to get $\boldsymbol{c}_z$. Since the encryption scheme is also homomorphic with respect to the randomness used for encryption, the hybrid arguments generated this way and real arguments are perfectly indistinguishable. Furthermore, since the Σ-protocol is SHVZK, hybrid arguments and simulated arguments are computationally indistinguishable. Furthermore, if the Σ-protocol has statistical SHVZK then the hybrid arguments and simulated arguments are statistically indistinguishable.

Finally, we will prove that the construction has adaptive culpable soundness with respect to R_{guilt}. Plugging our construction into the probability defining culpable soundness with a probabilistic polynomial time adversary $\mathcal{A}$ we get

$$\Pr\left[\begin{array}{l|l} (ek, dk) \leftarrow \mathcal{G}(1^n); e \leftarrow \{0,1\}^n; c \leftarrow \mathcal{E}_{ek}(e) & (x, w_{\text{guilt}}) \in R_{\text{guilt}} \\ (x, (a, \boldsymbol{c}_z), w_{\text{guilt}}) \leftarrow \mathcal{A}(ek, c); \boldsymbol{z} \leftarrow \mathcal{D}_{dk}(\boldsymbol{c}_z) & \mathcal{V}_\Sigma(x, a, e, \boldsymbol{z}) = 1 \end{array}\right].$$

By the unique identifiable challenge property of the Σ-protocol this probability is at most the chance that e is the unique answerable challenge:

$$\Pr\left[\begin{array}{l|l} (ek, dk) \leftarrow \mathcal{G}(1^n); e \leftarrow \{0,1\}^n; c \leftarrow \mathcal{E}_{ek}(e) & (x, w_{\text{guilt}}) \in R_{\text{guilt}} \\ (x, (a, \boldsymbol{c}_z), w_{\text{guilt}}) \leftarrow \mathcal{A}(ek, c); \boldsymbol{z} \leftarrow \mathcal{D}_{dk}(\boldsymbol{c}_z) & e = E(x, w_{\text{guilt}}, a) \end{array}\right].$$

By the IND-CPA security of the encryption scheme, this probability is at most negligibly larger than the same expression with c encrypting a random challenge e'

$$\Pr\left[\begin{array}{l|l} (ek, dk) \leftarrow \mathcal{G}(1^n); e, e' \leftarrow \{0,1\}^n; c \leftarrow \mathcal{E}_{ek}(e') & (x, w_{\text{guilt}}) \in R_{\text{guilt}} \\ (x, (a, \boldsymbol{c}_z), w_{\text{guilt}}) \leftarrow \mathcal{A}(ek, c); \boldsymbol{z} \leftarrow \mathcal{D}_{dk}(\boldsymbol{c}_z) & e = E(x, w_{\text{guilt}}, a) \end{array}\right].$$

Since e is chosen uniformly random this latter probability is at most 2^{-n}, which is negligible. □

3.1 Non-interactive Designated Verifier Arguments for Statements about Ciphertexts

In Sect. 2.2 we gave a Σ-protocol for proving a ciphertext having either 0 or 1 as plaintext. Using the DFN transformation, this leads to a non-interactive

designated verifier argument with culpable soundness for a ciphertext encrypting 0 or 1, i.e., for the relation

$$R = \left\{ ((ek, c), (m, r)) : m \in \{0,1\} \text{ and } r \in \{0,1\}^{\ell_r(n)} \text{ and } c = \mathcal{E}_{ek}(m; r) \right\}$$

with culpable soundness using

$$R_{\text{guilt}} = \left\{ ((ek, c), dk) : c \in \mathcal{C}_{ek} \text{ and } \mathcal{D}_{dk}(c) \notin \{0,1\} \text{ and } \text{VerifyKey}(1^n, ek, dk) = 1 \right\}.$$

This designated verifier argument works for ciphertexts produced by all strongly additively homomorphic encryption schemes that have message space $\mathbb{Z}_p$ for $p > 2^n$ such as for instance the Okamoto-Uchiyama [26] encryption scheme from Sect. 2.1. A second instance of the same strongly additively homomorphic encryption scheme but with larger message space can also be used for the DFN transformation. However, in the interest of more efficient implementations, it might be desirable to use a different encryption scheme for the DFN transformation. Specifically, DFN does not require the message space to be of prime order or the scheme to be *strongly* additively homomorphic, giving us the option of using an encryption scheme better suited for encrypting long messages such as Damgård-Jurik [12].

It is fairly simple to adapt standard Σ-protocols for other languages expressing properties about ciphertexts. In particular, in addition to the argument for encryption of 0 or 1 it is possible to construct non-interactive designated verifier arguments for the following relations:

Plaintext is 0: We can prove that a ciphertext c encrypts 0, i.e., give a non-interactive designated verifier argument for the relation

$$R^0 = \left\{ ((ek, c), r) : r \in \{0,1\}^{\ell_r(n)} \text{ and } c = \mathcal{E}_{ek}(0; r) \right\}.$$

Equivalence of plaintexts: Given two ciphertexts c and c', we can give a non-interactive designated verifier argument for them having the same plaintext by proving that c/c' is an encryption of 0 using the above designated verifier argument.

Multiplicative relationship: Given a triple of ciphertexts c_0, c_1 and c_2, we can prove that the plaintexts m_0, m_1 and m_2 satisfy $m_0 = m_1 m_2 \bmod p$. More precisely, we can construct a designated verifier argument for the relation

$$R^M = \left\{ \begin{array}{c} ((ek, c_0, c_1, c_2), (m_1, m_2, r_0, r_1, r_2)) : m_1, m_2 \in \mathbb{Z}_p, r_0, r_1, r_2 \in \{0,1\}^{\ell_r(n)} \\ c_0 = \mathcal{E}_{ek}(m_1 m_2; r_0) \text{ and } c_1 = \mathcal{E}_{ek}(m_1; r_1) \text{ and } c_2 = \mathcal{E}_{ek}(m_2; r_2) \end{array} \right\}.$$

In all cases, the corresponding guilt witness w_{guilt} consists of the decryption key, which can be used to decrypt the ciphertexts in the statement.

Circuit Satisfiability We will now show that given a circuit consisting of NAND-gates and encryptions of the wires it is possible to prove that the plaintexts

correspond to a satisfying assignment. A circuit C with $k+1$ wires and s gates can be described as $\{(j_1, j_2, j_3)\}_{j=1}^{s}$, which means that the wires should satisfy $w_{j_3} = \neg(w_{j_1} \wedge w_{j_2})$. We let the output wire be $w_0 = 1$ and the corresponding ciphertext be $c_0 = \mathcal{E}_{ek}(1;0)$ encrypted with randomness $r_0 = 0$. We consider the relations:

$$R^C = \left\{ \begin{array}{r} ((C, ek, c_1, \ldots, c_k), (w_1, r_1, \ldots, w_k, r_k)) \mid \forall j = 1, \ldots, s : w_{j_3} = \neg(w_{j_1} \wedge w_{j_2}) \\ \forall i = 1, \ldots, k : w_i \in \{0,1\} \wedge r_i \in \{0,1\}^{\ell_r(n)-2} \wedge c_i = \mathcal{E}_{ek}(w_i; r_i) \end{array} \right\},$$

$$R^C_{\text{guilt}} = \left\{ \begin{array}{l} ((C, ek, c_1, \ldots, c_k), dk) \mid \text{VerifyKey}(1^n, ek, dk) = 1 \text{ and } \forall i = 1, \ldots k : c_i \in \mathcal{C}_{ek} \\ \exists i \in \{1, \ldots, k\} : w_i = \mathcal{D}_{dk}(c_i) \notin \{0,1\} \text{ or } \exists j \in \{1, \ldots, s\} : w_{j_3} \neq \neg(w_{j_1} \wedge w_{j_2}) \end{array} \right\}.$$

The strategy in the designated verifier argument for R^C is to first prove that each ciphertext contains a wire value $w_i \in \{0,1\}$. Next, the prover proves for each NAND-gate (j_1, j_2, j_3) that $w_{j_3} = \neg(w_{j_1} \wedge w_{j_2})$. Following [21] we have for $w_{j_1}, w_{j_2}, w_{j_3} \in \{0,1\}$

$$w_{j_3} = \neg(w_{j_1} \wedge w_{j_2}) \quad \text{if and only if} \quad w_{j_1} + w_{j_2} + 2w_{j_3} - 2 \in \{0,1\}.$$

Using the homomorphic properties of the encryption scheme, we will therefore for each NAND-gate show $c_{j_1} c_{j_2} c_{j_3}^2 \mathcal{E}_{ek}(-2;0)$ contains 0 or 1. The full construction can be found in Fig. 4

$\mathcal{P}^C(pk, (C, ek, c_1, \ldots, c_k), (w_1, r_1, \ldots, w_k, r_k))$	$\mathcal{V}^C(vk, (C, ek, c_1, \ldots, c_k), \pi)$
$w_0 = 1, r_0 = 0, c_0 = \mathcal{E}_{ek}(w_0; r_0)$	$w_0 = 1, r_0 = 0, c_0 = \mathcal{E}_{ek}(w_0; r_0)$
For $i = 1, \ldots, k$	Parse $C = \{(j_1, j_2, j_3)\}_{j=1}^{s}$
$\quad \pi_i \leftarrow \mathcal{P}(pk, c_i, (w_i, r_i))$	For $j = 1, \ldots, s$
Parse $C = \{(j_1, j_2, j_3)\}_{j=1}^{s}$	$\quad c'_j = c_{j_1} c_{j_2} c_{j_3}^2 \mathcal{E}_{ek}(-2;0)$
For $j = 1, \ldots, s$	Parse $\pi = (\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_s)$
$\quad c'_j = c_{j_1} c_{j_2} c_{j_3}^2 \mathcal{E}_{ek}(-2;0)$	Accept if and only if
$\quad m'_j = w_{j_1} + w_{j_2} + 2w_{j_3} - 2$	$\quad$ For $i = 1, \ldots, k$
$\quad r'_j = r_{j_1} + r_{j_2} + 2r_{j_3}$	$\quad\quad \mathcal{V}(vk, c_i, \pi_i) = 1$
$\quad \pi'_j \leftarrow \mathcal{P}(pk, c'_j, (m'_j, r'_j))$	$\quad$ For $j = 1, \ldots, s$
Return $\pi = (\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_s)$	$\quad\quad \mathcal{V}(vk, c'_j, \pi'_j) = 1$

Fig. 4. Non-interactive designated verifier argument $(G^C, \mathcal{P}^C, \mathcal{V}^C)$ for encryption of satisfying assignment of wires in a circuit using $\mathcal{G}^C = \mathcal{G}$ where $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a designated verifier argument for encryption of 0 or 1.

Theorem 3. *$(\mathcal{G}^C, \mathcal{P}^C, \mathcal{V}^C)$ given in Fig. 4 is a non-interactive designated verifier argument for R^C with culpable soundness using R^C_{guilt} if $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a non-interactive designated verifier argument for encryption of 0 or 1 using R_{guilt} from Sect. 2.2.*

Proof. Perfect completeness follows from the homomorphic properties of the encryption scheme and the perfect completeness of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$.

We will now prove composable zero-knowledge. The simulator $\mathcal{S}^C(vk, (C, ek, c_1, \ldots, c_k))$ runs like the prover except it simulates the proofs $\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_s$ as $\pi_i \leftarrow \mathcal{S}(vk, (ek, c_i))$ and $\pi'_j \leftarrow \mathcal{S}(vk, c'_j)$. A straightforward hybrid argument shows that this is indistinguishable from a real proof.[3]

Finally, we will prove that the argument is culpably sound. By the culpable soundness of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ when we are given dk by the adversary, the proofs $\pi_1, \ldots, \pi_k$ guarantee each ciphertext contains 0 or 1. The homomorphic property of the encryption scheme combined with the culpable soundness of the proofs $\pi'_1, \ldots, \pi'_s$ then shows that the plaintexts respect the NAND-gates. Since the output is $w_0 = 1$ this means the circuit is satisfied by the encrypted values. □

4 Applications in Voting with Homomorphic Tallying

We will use a basic referendum voting scheme as an illustration of how to use non-interactive zero-knowledge designated verifier arguments with culpable soundness. We use a modification of the framework by Bernhard et al. [4] which generalises the Helios voting system. Such schemes operate by having eligible voters post their votes on a bulletin board encrypted with an additively homomorphic encryption scheme. The election result can then be produced by a single decryption operation on the homomorphic sum of the individual votes. Zero-knowledge protocols ensure that the various participating parties remain honest.

4.1 Voting Schemes

We assume a bulletin board BB holds all messages posted by the various participants in the election, and that it behaves honestly for the entirety of the election. During the submission of ballots, it operates in an append-only mode *without* disclosing its contents. After voting has concluded, the bulletin board reveals the ballots it contains and checks their validity. The checks use only public information and as such are reproducible by any party; the bulletin board performs them for convenience. Finally, we assume that the history of the bulletin board is publicly accessible as well as the current state.

Our use of a *delayed* bulletin board is a departure from usual practice and is aimed at preventing attacks based on malleability. The additional trust placed on the board by this requirement may be mitigated by having the bulletin board immediately display commitments to ballots or eliminated by augmenting the ballot encryption to be submission secure [29]. We also note that Cortier et al. [7] develop techniques to guard against misbehaving boards.

In the interest of simplicity, we restrict the options in the referendum to $\{0, 1\}$ without giving the option of casting an abstention ballot. The election is run by

[3] We remark that here the usefulness of *composable* zero-knowledge comes into play since the hybrid arguments are indistinguishable even to an adversary with access to the verification key vk, which allows the hybrid argument to go through.

two trustees, $\mathcal{T}_D$ and T_V, tasked with holding the decryption and verification keys for the election. We will for simplicity consider them to be trusted parties but they could be implemented using threshold cryptography.

Definition 8 (Voting Scheme). *A voting scheme Π consists of five probabilistic polynomial time algorithms:* **Setup**, **Vote**, **SubmitBallot**, **CheckBoard**, **Tally**, *which operate as follows:*

Setup *The setup algorithm takes as input a security parameter 1^n. It produces secret information SEC, public information PUB and verification information AUG. It also initialises the bulletin board BB and sets it to be hidden. PUB is assumed to be public knowledge after* **Setup** *has run.*

Vote *Vote accepts a vote $m \in \{0,1\}$ and outputs a ballot B encoding m.*

SubmitBallot *SubmitBallot(B, BB) takes as input a ballot B and the current state of the bulletin board BB and outputs either $(0, BB)$ if it rejects B or $(1, BB \xleftarrow{+} B)$ if it accepts it.*

CheckBoard *CheckBoard(BB, AUG) makes BB visible, and then checks all ballots on BB, replacing with $\perp$ any ballots that do not pass the verification tests. After checking, the verification information of valid ballots can be removed from the board.*

Tally *Tally (BB, SEC) takes as input a verified bulletin board BB and the secret information SEC and outputs the election result.*

For correctness we require that the ballots of honest voters are counted correctly, and that ballots cast by malicious voters cannot influence the election more than an honest one (i.e casting q malicious ballots can only add q votes and subtract none).

$\mathbf{Exp}^{COR}_{\Pi,\mathcal{A}}(n)$
$(PUB, SEC, AUG) \leftarrow \mathbf{Setup}(1^n)$
$(vsum, q) \leftarrow (0, 0)$
$\mathcal{A}^{\mathbf{VoteOracle}(\cdot),\mathbf{BallotOracle}(\cdot)}(PUB)$
$BB \leftarrow \mathbf{CheckBoard}(BB, AUG)$
$result \leftarrow \mathbf{Tally}(BB, SEC)$
Return $(result, vsum, q)$

VoteOracle(v)
$B \leftarrow \mathbf{Vote}(v)$
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
if $r = accept : vsum \leftarrow (vsum + v)$
Return (r, B)

BallotOracle(B)
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
$q \leftarrow q + 1$
Return r

Fig. 5. The referendum correctness experiment, and the oracles provided to the adversary.

Definition 9 (Correctness). *We say that a referendum voting scheme Π is correct if for all efficient adversaries $\mathcal{A}$:*

$$\Pr\left[(result, vsum, q) \leftarrow \mathbf{Exp}^{COR}_{\Pi,\mathcal{A}}(n) : vsum \leq result \leq q + vsum\right] \approx 1$$

Definition 10 (Ballot Privacy). *We say that a voting scheme Π satisfies ballot privacy if for all efficient stateful interactive adversaries $\mathcal{A}$:*

$$\Pr\left[\mathbf{Exp}^{BP}_{\Pi,\mathcal{A}}(n) = 1\right] \approx \frac{1}{2}$$

$\mathbf{Exp}^{BP}_{\Pi,\mathcal{A}}(n)$

$(PUB, SEC, AUG) \leftarrow \mathbf{Setup}(1^n)$
$b \leftarrow \{0,1\}$
$\mathcal{A}^{\mathbf{VoteOracle}(\cdot),\mathbf{BallotOracle}(\cdot)}(PUB)$
$BB \leftarrow \mathbf{CheckBoard}(BB, AUG)$
$BB' \leftarrow \mathbf{CheckBoard}(BB', AUG)$
$result \leftarrow \mathbf{Tally}(BB', SEC)$
$\hat{b} \leftarrow \mathcal{A}(result, BB, AUG)$
Return $b = \hat{b}$

VoteOracle(v)

$B' \leftarrow \mathbf{Vote}(v)$
if $b = 1$ then $B \leftarrow B'$
  else $B \leftarrow \mathbf{Vote}(0)$
$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
$(r', BB') \leftarrow \mathbf{SubmitBallot}(B', BB')$
Return (r, B)

BallotOracle(B)

$(r, BB) \leftarrow \mathbf{SubmitBallot}(B, BB)$
if $r = accept$ then
  $(r', BB') \leftarrow \mathbf{SubmitBallot}(B, BB')$
Return r

Fig. 6. The Ballot Privacy experiment, and the oracles provided to the adversary.

4.2 A Referendum Voting Scheme

We will now describe a voting scheme Π^{REF} for a yes-no referendum, based on an additively homomorphic encryption scheme such as $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ and with a non-interactive designated verifier argument system $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for a plaintext being 0 or 1 such as the one given in Fig. 3.

For simplicity, we omit correctness proofs for keys having been generated correctly but point out that since the setup involves a limited number of parties we could assume the use of online zero-knowledge protocols using standard techniques. We also assume that the bulletin board behaves honestly. We now give descriptions of the Voting Protocol Algorithms:

Setup The setup algorithm takes as input a security parameter 1^n. The decryption trustee $\mathcal{T}_D$ runs $K(1^n)$ to produce (ek, dk) and the verification trustee then runs $\mathcal{G}(1^n)$ to obtain (pk, vk). Let, $PUB = (ek, pk)$, $AUG = vk$ and $SEC = dk$. The procedure also initialises the bulletin board BB to be hidden, and publishes PUB.

Vote(m) Pick $r \leftarrow \{0,1\}^{\ell_r(n)}$ and return (c, π), where $c = \mathcal{E}_{ek}(m; r)$ and $\pi \leftarrow \mathcal{P}(pk, (ek, c, r))$.

SubmitBallot(B,BB) Return $(accept, BB \xleftarrow{+} B)$.

CheckBoard(BB,AUG) The bulletin board BB becomes visible. $\mathcal{T}_V$ publishes AUG. For every ballot $B = (c, \pi)$ in BB we check whether $c \in \mathcal{C}_{ek}$ and $\mathcal{V}(vk, (ek, c), \pi) = 1$. If not, they will be omitted from the tally.

Tally(BB,SEC) The decryption trustee publishes $result = \mathcal{D}_{dk}(\prod_{i=1}^{k} c_i)$, where $c_1, \ldots, c_k$ are the encrypted votes that passed the validity check.

Theorem 4. *The referendum scheme Π^{REF} defined above is correct.*

Proof. Let $\mathcal{A}$ be an adversary against $\mathbf{Exp}^{COR}_{\Pi,\mathcal{A}}(n)$ that causes *result* to be out of bounds with non-negligible probability. We construct a simulator $\mathcal{B}$ that contradicts the adaptive culpable soundness of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. $\mathcal{B}$ will simulate the correctness experiment for $\mathcal{A}$ while acting as the adversary for the adaptive culpable soundness experiment. $\mathcal{B}$ operates by running the correctness experiment normally with the difference that it does not generate (pk, vk) but instead obtains pk from the adaptive culpable soundness experiment.

Because $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is correct and additively homomorphic, *result* being out of bounds implies one of the submitted ballots $B = (c, \pi)$ is such that c encrypts a value other than 0 or 1 while at the same time $\mathcal{V}(vk, (ek, c), \pi) = 1$. Choosing one of the q ballots at random, $\mathcal{B}$ outputs $(x, \pi, w_{\text{guilt}})$ to the experiment, where $x = (c, pk)$ and $w_{\text{guilt}} = dk$. Since q is polynomial in n this gives $\mathcal{B}$ a non-negligible probability of winning the experiment. □

Theorem 5. *The scheme Π^{REF} satisfies ballot privacy.*

Proof. We will prove that $\mathcal{A}$ can not do better than guess the value of b in the ballot privacy experiment via a series of hybrid games. We exploit the fact that the $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ argument system achieves statistical zero-knowledge, the fact that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is IND-CPA secure as well as the delay on the bulletin board. We also take advantage of the fact that in a referendum the number of possible results is linear in the number of votes.

We will focus on the **VoteOracle** calls that the adversary makes, as that is where the experiment diverges depending on b. Let q_v,q_b be upper bounds on the number of **VoteOracle** and **BallotOracle** queries made by $\mathcal{A}$ for a particular security parameter n. Let $q_\Sigma = q_v + q_b$ be the total number of queries.

We define as $\mathbf{Exp}^2$ the experiment $\mathbf{Exp}^{BP}_{\Pi^{REF},\mathcal{A}}$ where all **VoteOracle** calls produce a ballot with a simulated proof π instead of a real one. We also define a series of hybrid games H^1_i for $i \in \{0, q_v\}$ in which the first i **VoteOracle** calls produce a ballot with a simulated proof π instead of a real one. Via a straightforward hybrid argument, if $\mathcal{A}$ can distinguish between $\mathbf{Exp}^2 = H^1_{q_v}$ and $\mathbf{Exp}^{BP}_{\Pi^{REF},\mathcal{A}} = H^1_0$ with a non-negligible probability there must be a value of i such that he can distinguish H^1_i and H^1_{i+1}. This contradicts the honest verifier zero knowledge property of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, since for all i, H^1_i and H^1_{i+1} differ (at most)

only in a single proof transcript. Thus $\mathcal{A}$ wins $\mathbf{Exp}^2$ with probability negligibly close to $\mathbf{Exp}^{BP}_{\Pi^{REF},\mathcal{A}}$.

We also define a series of hybrid games H^2_i for $i \in \{0, q_v\}$ as $\mathbf{Exp}^2$ in which the first i **VoteOracle** calls operate as if $b = 1$ and the rest as if $b = 0$. If $\mathcal{A}$ can win $\mathbf{Exp}^2$ with non-negligible probability, he can distinguish between H^2_0 and $H^2_{q_v}$ and thus there must be a value of i such that $\mathcal{A}$ can distinguish H^2_i and H^2_{i+1}.

Let the variable RES be the sum of the votes contained in ballots with correct proofs which appear in the bulletin board BB' before **CheckBoard** is called. We note that RES only takes values in $\{0, \ldots, q_\Sigma\}$.

Let $p(n)$ be a polynomial such that $\mathcal{A}$ can distinguish between H^2_i and H^2_{i+1} with probability at least $\frac{1}{2} + \frac{1}{p(n)}$ for an infinite number of $n \in \mathbb{N}$. We will construct an adversary $\mathcal{B}$ that can obtain non-negligible advantage against the IND-CPA security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ by simulating an election against $\mathcal{A}$. Initially $\mathcal{B}$ obtains a public key ek from the IND-CPA experiment and completes the setup as normal, obtaining (pk, vk) from $\mathcal{G}$. $\mathcal{B}$ proceeds by following Π^{REF} with the following difference: the first i **VoteOracle** calls operate as if $b = 1$. The next **VoteOracle** (which we can assume w.l.o.g to be $v^* = 1$) is answered as if it was successful, but $\mathcal{B}$ does not update BB. Afterwards, before **CheckBoard** is called, the experiment is suspended and the state st of $\mathcal{A}$ is saved along with the bulletin boards and keys as $\sigma = (ek, (pk, vk), st, BB, BB')$. $\mathcal{B}$ does not know the value of RES, but knows that it takes values in $\{0, \ldots, q_\Sigma\}$.

Let $\mathbf{Exp}^3(\sigma, v, r)$, where $\sigma = (ek, (pk, vk), st, BB, BB')$ be the following experiment: Produce B^* as a fresh ballot (with simulated proof) containing v, add B^* to BB, restore the adversary's state to st and resume the voting protocol starting at **CheckBoard**. The **Tally** query is answered with r. The result of the experiment is $v == \hat{b}$ where $\hat{b}$ is $\mathcal{A}$'s reply.

We note when $r = RES$, $\mathbf{Exp}^3(\sigma, v, r)$ produces the same output as H^2_i or H^2_{i+1} depending only on v. We call a saved state σ "good" if $\mathcal{A}$ has a non-negligible advantage in distinguishing $\mathbf{Exp}^3(\sigma, 0, RES)$ from $\mathbf{Exp}^3(\sigma, 1, RES)$, where RES is determined uniquely by BB'. Because $\mathcal{A}$ can distinguish between H^2_i and H^2_{i+1}, a state created by $\mathcal{B}$ must be "good" with non-negligible probability.

$\mathcal{B}$ repeats the following $n \cdot p(n)$ times: run $\mathbf{Exp}^3(\sigma, v, r)$ for all combinations of $v \in \{0, 1\}$ and $r \in \{0, \ldots, q_\Sigma\}$.

Afterwards, $\mathcal{B}$ can determine the value of r for which $\mathcal{A}$ has best distinguished between $v = 0$ and 1. We note that because the true value of RES is included in the iterations, if σ is a "good" state, there is a value of r for which $\mathcal{A}$ achieves at least $\frac{1}{p(n)}$ advantage in distinguishing v, and after $n \cdot p(n)$ experiments Chernoff-bounds show that $\mathcal{B}$ has a good estimate of $\mathcal{A}$'s advantage for each value of r.

After determining the optimal value of r, $\mathcal{B}$ will send $(0, 1)$ to the IND-CPA experiment and obtain a challenge ciphertext $\hat{c}$. $\mathcal{B}$ produces a simulated proof $\hat{\pi}$ for c, adds $\hat{B}$ to the saved board and resumes the experiment a final time. $\mathcal{B}$ finally forwards the reply of A to the IND-CPA experiment.

Because $\mathcal{B}$ will proceed without restarting with non-negligible probability, it runs in expected polynomial time. Because with overwhelming probability $\mathcal{B}$ proceeds only when σ is a "good" state, the advantage in distinguishing whether c contains 0 or 1 is non-negligible. □

References

1. Ben Adida. Helios: Web-based open-audit voting. In *Security Symposium*, SS'08, pages 335–348. USENIX Association, 2008.
2. Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *Foundations of Computer Science*, FOCS '04, pages 186–195. IEEE, 2004.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Computer and Communications Security*, pages 62–73. ACM, 1993.
4. David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting Helios for provable ballot privacy. In *Computer Security–ESORICS 2011*, pages 335–354. Springer, 2011.
5. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Theory of Computing*, STOC '88, pages 103–112. ACM, 1988.
6. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.
7. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Distributed ElGamal á la Pedersen: Application to Helios. In *Privacy in the Electronic Society*, WPES '13, pages 131–142. ACM, 2013.
8. Ronald Cramer and Victor Shoup. A practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO'98*, pages 13–25. Springer, 1998.
9. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology—EUROCRYPT 2002*, pages 45–64. Springer, 2002.
10. Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *Advances in cryptology—EUROCRYPT'92*, pages 341–355. Springer, 1992.
11. Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In *Theory of Cryptography*, pages 41–59. Springer, 2006.
12. Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.
13. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1987.
14. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Advances in Cryptology—EUROCRYPT 2013*, pages 626–645. Springer, 2013.
15. Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, pages 1–24, 2014.

16. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Foundations of Computer Science*, FOCS '03, pages 102–113. IEEE, 2003.
17. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.
18. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology–ASIACRYPT 2006*, pages 444–459. Springer, 2006.
19. Jens Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology–ASIACRYPT 2010*, pages 341–358. Springer, 2010.
20. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology–ASIACRYPT 2010*, pages 321–340. Springer, 2010.
21. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM*, 59(3):11:1–11:35, 2012.
22. Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM Journal on Computing*, 41(5):1193–1232, 2012.
23. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology—EUROCRYPT'96*, pages 143–154. Springer, 1996.
24. Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.
25. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Theory of Computing*, STOC 2013, pages 427–437. ACM, 1990.
26. Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in cryptology—EUROCRYPT'98*, pages 308–318. Springer, 1998.
27. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Security and Privacy*, pages 238–252. IEEE, 2013.
28. Carmine Ventre and Ivan Visconti. Co-sound zero-knowledge with public keys. In *Progress in Cryptology–AFRICACRYPT 2009*, pages 287–304. Springer, 2009.
29. Douglas Wikström. Simplified submission of inputs to protocols. In *Security and Cryptography for Networks*, pages 293–308. Springer, 2008.