Efficient Proof of Vote Validity Without Honest-Verifier Assumption in Homomorphic E-Voting

Kun Peng*

Abstract—Vote validity proof and verification is an efficiency bottleneck and privacy drawback in homomorphic e-voting. The existing vote validity proof technique is inefficient and only achieves honest-verifier zero knowledge. In this paper, an efficient proof and verification technique is proposed to guarantee vote validity in homomorphic e-voting. The new proof technique is mainly based on hash function operations that only need a very small number of costly public key cryptographic operations. It can handle untrusted verifiers and achieve stronger zero knowledge privacy. As a result, the efficiency and privacy of homomorphic e-voting applications will be significantly improved.

Keywords—Efficient Proof, E-Voting

1. INTRODUCTION

E-voting is a popular application of cryptology to secure network services. It is well known that it can exploit homomorphism of the so-called additive homomorphic encryption algorithms to protect the privacy of the voters. An encryption algorithm with decryption function $D()$ is additive homomorphic if $D(c_1c_2) = D(c_1) + D(c_2)$ for any ciphertexts c_1 and c_2. In secure e-voting, sum of the votes encrypted with an additive homomorphic encryption algorithm can be recovered without revealing any single vote. Such e-voting schemes are called homomorphic e-voting [1, 2, 5, 9, 11-16, 19-21], which count the votes without decrypting their encryptions. In homomorphic e-voting, a voter has to include a voting choice for every candidate in his vote. If the voter wants to elect a candidate, he encrypts a "1" choice for him; otherwise he encrypts a "0" for the candidate. Then the votes are encrypted using an additive homomorphic encryption algorithm. The talliers exploit homomorphism of the encryption algorithm to find out sum of the votes for every candidate without decrypting any single vote. In homomorphic e-voting, each voting choice must be either 1 or 0 to guarantee correctness of the homomorphic tallying. So validity of the votes must be proved by the voters and then publicly verified. This proof and verification is highly inefficient in the existing homomorphic e-voting schemes [1, 2, 5, 9, 11-16, 19-21]. They usually employ ZK (zero knowledge) proof of partial knowledge [4] and need a cost linear in the number of candidates for every vote.

Recently, a couple of schemes [17, 18] are proposed to improve efficiency of homomorphic e-voting. However, they are not general techniques suitable for most homomorphic e-voting applications. The homomorphic e-voting scheme in [17] can only handle a small number of voters. In most election applications, the number of voters overflows its upper limit and their

Manuscript received April 27, 2011; accepted June 30, 2011.
Corresponding Author: Kun Peng
* Institute for Inforcomm Research, Singapore (dr.kun.peng@gmail.com)

votes must be grouped and tallying must be separately carried out in every group. The homomorphic e-voting schemes in [18] is only efficient when there is only one verifier to check validity of the votes[1]. So it is not suitable for publicly verifiable e-voting applications, where there are many verifiers including the talliers and other observers. As we study general efficient vote validity checking mechanism in this paper, the techniques in [17, 18] are incomparable to our work.

Another drawback of the vote validity proof techniques in the existing homomorphic e-voting schemes including [1, 2, 5, 9, 11-16, 19-21] and [17, 18] is that they employ the so-called honest verifier ZK proof security model such that their privacy depends on a trust assumption that the verifiers in the proof are honest. So, in their interactive form they require that the verifiers randomly choose some challenges and in their non-interactive form (except for the vote validity proof protocol in [18], which cannot be non-interactive) they depend on the random oracle model.

In this paper, an efficient vote validity proof mechanism including three proof protocols is proposed. It is mainly based on efficient hash function operations and symmetric cipher operations and only a small number of public key cryptographic operations are employed. So it is more efficient in both computation and communication than the existing vote validity proof techniques. Moreover, it upgrades honest verifier ZK to handle dishonest verifiers in the proof and can guarantee privacy even if the verifier is not trusted. Using our new proof technique, homomorphic e-voting can be not only more efficient to support users with limited computational capability and communicational bandwidth (e.g., mobile users) but also more private to support election applications with strong security requirements.

2. BACKGROUND

Technical background about homomorphic e-voting and security models for its vote validity proof are provided in this section.

2.1 Zero Knowledge Model in Privacy Analysis

Security of proof protocols are usually analysed using a formal method called ZK (zero knowledge) defined as follows.

Definition 1: (Zero Knowledge Proof) A zero knowledge proof can be formally denoted as $PR(\alpha_1,\alpha_2,\ldots,\alpha_\zeta \mid \phi_1(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_1,\phi_2(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_2,\ldots,\phi_n(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_n)$ where $\alpha_1,\alpha_2,\ldots,\alpha_\zeta$ stand for the prover's secret knowledge, $\beta_1,\beta_2,\ldots,\beta_n$ stands for some public commitments to the secrets and functions and $\phi_1(),\phi_2(),\ldots,\phi_n()$ describe the properties of the language.

Definition 2: (Completeness of zero knowledge proof) In a complete zero knowledge proof protocol $PR(\alpha_1,\alpha_2,\ldots,\alpha_\zeta \mid \phi_1(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_1,\phi_2(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_2,\ldots,\phi_n(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_n)$, if the prover knows $\alpha_1,\alpha_2,\ldots,\alpha_\zeta$ to satisfy $\phi_1(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_1,\phi_2(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_2,\ldots,\phi_n(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_n$, he can always pass the verification.

Definition 3: (Soundness of zero knowledge proof) If a sound zero knowledge proof $PR(\alpha_1,\alpha_2,\ldots,\alpha_\zeta \mid \phi_1(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_1,\phi_2(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_2,\ldots,\phi_n(\alpha_1,\alpha_2,\ldots,\alpha_\zeta)=\beta_n)$ passed its veri-

fication with a non-negligible probability, it is guaranteed that the prover can calculate in polynomial time $\alpha_1, \alpha_2, \ldots, \alpha_\zeta$ to satisfy $\phi_1(\alpha_1, \alpha_2, \ldots, \alpha_\zeta) = \beta_1, \phi_2(\alpha_1, \alpha_2, \ldots, \alpha_\zeta) = \beta_2, \ldots, \phi_n(\alpha_1, \alpha_2, \ldots, \alpha_\zeta) = \beta_n$.

Definition 4: (Privacy of zero knowledge proof) No information about the secret integers is revealed in the proof protocol. More precisely, there exists a polynomial algorithm without any access to the secret integers to generate a simulating proof transcript with the same distribution as the real proof transcript.

Privacy of ZK proof can be handled in different models. The simplest model assumes that the verifier is honest and does not try to extract any secret from the prover by deviating from the proof protocol. Zero knowledge in this model is called "honest-verifier zero knowledge." Although many proof protocols only achieve honest-verifier zero knowledge and are appreciated and employed in many applications, we still want to handle dishonest verifier and achieve stronger privacy in our proof technique. Although there are some costly techniques (e.g., [6]) to upgrade honest-verifier zero knowledge to perfect zero knowledge, we prefer more efficient solution to achieve trust-free privacy and design a vote validity proof protocol to tolerate an untrusted verifier.

2.2 Homomorphic E-Voting

Homomorphic e-voting schemes [1, 2, 5, 9, 11-16, 19-21] employ an additive homomorphic encryption algorithm with a distributed decryption function to seal the votes. The tallying operation in homomorphic e-voting decrypts the sum of the votes instead of the separate votes. So it only costs one single decryption operation to count the votes for each candidate and is quite efficient. However, correctness of homomorphic voting depends on validity of the votes. An invalid vote can be the sum of multiple valid votes to compromise correctness of a homomorphic voting scheme. So, a vote validity check mechanism is necessary to detect and delete any invalid vote before the tallying phase.

A typical additive homomorphic encryption algorithm with a *T*-out-of-*M* distributed decryption function is distributed Paillier encryption proposed by Fouque et al [8], which is recalled as follows.

2.2.1 Key generation:

$N = pq$, $p = 2p' + 1$ and $q = 2q' + 1$ where p and q are primes and $gcd(N, \varphi(N)) = 1$. Integers a and b are randomly chosen from Z_N^* and $g = (1+N)^a + b^N \bmod N^2$. The private key is $\beta p'q'$ where β is randomly chosen from Z_N^*. The public key consists of N, g and $\theta = a\beta p'q'$. $A_1, A_2, \ldots, A_M$ are the private key holders. Let $F(x) = \sum_{k=0}^{T-1} f_k x^k$ where $f_0 = \beta p'q'$ and f_1, f_2, f_{T-1} are random integers in $Z_{p'q'}$. The share $d_j = F(j) \bmod p'q'N$ is distributed to A_j for $j = 1, 2, \ldots, M$. G is the cyclic subgroup containing all the quadratic residues in $Z_{N^2}^*$ where an integer y in Z_{N^2} is an N^{th} residue if there exist an integer x such that $x^N = y \bmod N^2$. Random integer v is a generator of G and $v_j = v^{\Delta d_j} \bmod N^2$ for *j=1,2,...M* where $\Delta = M!$. Integers v and v_j for *j=1,2,...M* are published.

2.2.2 Encryption:

A message $s \in Z_N$ is encrypted into $c = g^s r^N \bmod N^2$ where r is randomly chosen from Z_N^*.

2.2.3 Partial decryptions of ciphertext c:

For *j=1,2,...M*, A_j each private key share holder provides his part of decryption $d_j = c^{2\Delta d_j}$ and proves $\log_{c^{4\Delta}} d_j^2 = \log_{v^{\Delta}} v_j$.

2.2.4 Combination of partial decryptions:

The final decryption result can be recovered as $s = L(\prod_{j\in\Psi} d_j^{2u_i}) \times \frac{1}{4\Delta^2\theta}$ where set Ψ contains the indices of *T* correct partial decryptions and $u_j = \Delta \prod_{1\le j'\le M, j'\ne j} \frac{j'}{j'-j}$.

The complete decryption function in the distributed Paillier encryption, including the partial decryptions and the combination, is denoted as *D( )*. With this encryption algorithm to seal the votes, most of the existing homomorphic e-voting schemes can be abstracted into the following protocol.

1. Suppose there are *n* voters and *w* candidates and each voter has to elect some of the candidates in his vote. It is required that *n*<*N*, which is satisfied in any practical election.

2. Each voter V_i chooses his voting vector $(s_{i,1}, s_{i,2}, \ldots, s_{i,w})$ where $s_{i,l} = 0$ or $s_{i,l} = 1$ for *l=1,2,...w*. A rule is followed: $s_{i,l} = 1$ if the l^{th} candidate is V_i election.

3. The distributed Paillier encryption recalled above is employed to encrypt the votes where the private key is shared among talliers $A_1, A_2, \ldots, A_M$. Each vote $(s_{i,1}, s_{i,2}, \ldots, s_{i,w})$ is encrypted into $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ where $c_{i,l} = g^{s_{i,l}} r_{i,l}^N \bmod N^2$ and $r_{i,l}$ is randomly chosen from Z_N^* for *l=1,2,...w*.

4. Each voter V_i illustrates the validity of his vote through proof of

$$KN[\, c_{i,l}^{1/N} \,] \vee KN[\, (c_{i,l}/g)^{1/N} \,] \; for\; l = 1,2,\ldots,w \tag{1}$$

$$and \qquad KN[\, ((\prod_{1}^{w} c_{i,l})/g)^{1/N} \,] \tag{2}$$

where *KN(x)* denotes knowledge of x. Proof of (1) is implemented by running the proof protocol Fig 1 (which is a combination of ZK proof of knowledge of N^{th} root [10] and ZK proof of partial knowledge [4]) for l=1,2,…w and Proof of (2) is a simple ZK proof of knowledge of N^{th} root.

5. The talliers verify the voters' proof of validity of their votes. The votes failing to pass the verification are deleted.

6. For simplicity, assume that all the votes pass the vote validity check. As *n*<*N* and the votes are valid, *T* honest talliers among $A_1, A_2, \ldots, A_M$ cooperate to decrypt $\prod_{i=1}^{n} c_{i,l}$ into the number of votes for the l^{th} candidate for *l=1,2,...w*. After the decryption, each A_j publicly proves that his partial decryption is correct.

7. The candidate with the most votes wins.

1. V_i publishes $a_{s_{i,l}} = r^N \bmod N^2$

$$a_{1-s_{i,l}} = u_{1-s_{i,l}}^N / (c_{i,l}/g^{1-s_{i,l}})^{v_{1-s_{i,l}}} \bmod N^2$$

where $r \in Z_N^*$, $v_{1-s_{i,l}} \in Z_N$, and $u_{1-s_{i,l}} \in Z_N^*$ are randomly chosen.

2. An integer v is randomly chosen from Z_N by a verifier or a (pseudo)random function.
3. V_i publishes u_0, u_1, v_0 and v_1 where

$$u_{s_{i,l}} = r s_{i,l}^{v_{s_{i,l}}} \bmod N^2$$
$$v_{s_{i,l}} = v - v_{1-s_{i,l}} \bmod Z_N$$

Public verification:

$$u_0^N = a_0 c_{i,l}^{v_0} \bmod N^2$$
$$u_1^N = a_1 (c_{i,l}/g)^{v_1} \bmod N^2$$
$$v = v_0 + v_1 \bmod Z_N$$

Fig. 1. Proof and verification of (1) to be repeated w times

Completeness and soundness of ZK proof of knowledge of the N^{th} root [10] and ZK proof of partial knowledge [4] guarantee that with an overwhelmingly large probability a voter's vote is valid iff his proof of validity of his vote can pass public verification. Honest verifier ZK property of ZK proof of knowledge of root [10] and ZK proof of partial knowledge [4] guarantees that no vote is revealed in a vote validity check. Tallying is very efficient and only costs each tallier *3w* exponentiations. Vote validity check is inefficient as it repeats the proof and verification protocol in Fig 1 for w times. The vote validity check brings each voter even higher a cost than vote encryption and a verifier at least O(wn) exponentiations. So vote validity check is the efficiency bottleneck of homomorphic e-voting.

3. EFFICIENT AND GENERAL VOTE VALIDITY PROOF

In this section, a new vote validity proof technique is designed. Firstly, it needs to slightly modify Paillier encryption algorithm to encrypt the votes in a special way. Then it reduces validity proof of a specially encrypted vote to proof of knowledge of multiple discrete logarithms, which is proved by a novel ZK proof without any trust assumption.

3.1 Adjusting Vote Encryption and Reducing Vote Validity Proof to Simple Zero Knowledge Proof

As explained in Section 2, the vote $(s_{i,1}, s_{i,2}, \ldots, s_{i,w})$ is a vector containing one 1 and *w-1* 0s. In the following, we show how to reduce validity proof of such a vote to simple proofs, where vote encryption is slightly adjusted to support the reduction.

1. Each V_i encrypts $(s_{i,1}, s_{i,2}, \ldots, s_{i,w})$ into $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ where $c_{i,l} = g^{s_{i,l}} R_i^{r_{i,l}N} \bmod N^2$, R_i is randomly chosen from Z_N^* and $r_{i,l}$ is randomly chosen from Z_N for *l=1,2,...w*. All the encrypted voting vectors are submitted to the talliers.

2. The talliers $A_1, A_2, \ldots, A_M$ cooperate to generate a public set $S = \{s_1, s_2, \ldots, s_w\}$, where each s_l is a random integer in Z_K corporately chosen by all the talliers. K is a large security pa-

rameter no larger than N but guaranteeing that w/K is negligible. For example, they can generate S as follows.

(a) Each A_j randomly chooses $S_{l,j}$ in Z_K and publishes $H_{l,j} = H_j(S_{l,j})$ for $l=1,2,...w$ where $H_j()$ is a one-way and collision-resistent hash function.

(b) After all the $H_{l,j}$s are published, each A_j publishes $S_{l,j}$ for $l=1,2,...w$ and anyone can verify their validity against their commitments $H_{l,j}$ for $l=1,2,...w$.

(c) $s_l = \sum_{j=1}^{M} S_{l,j} \bmod K$ for $l=1,2,...w$.

3. Each V_i has to prove that $C_i = \prod_{l=1}^{w} c_{i,l}^{s_l} \bmod N^2$ encrypts a message in S. As illustrated in Theorem 1, this proof guarantees that the voting vector encrypted in $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ is valid with an overwhelmingly large probability.

Theorem 1: If the voting vector in $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ is invalid, the probability that the message encrypted in C_i lies in S is negligible.

Proof: As $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ is invalid, there are the following two possibilities where $(s_{i,1}, s_{i,2}, \ldots, s_{i,w})$ is the voting vector encrypted into $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$.

- There is only one non-zero integer in $s_{i,1}, s_{i,2}, \ldots, s_{i,w}$.
- There are more than one non-zero integers in $s_{i,1}, s_{i,2}, \ldots, s_{i,w}$.

In the first case, suppose $s_{i,l'} \neq 0$. As $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ is invalid, $s_{i,l'} \neq 1 \bmod N$. So

$$D(C_i) = D(\prod_{l=1}^{w} c_{i,l}^{s_l}) = \sum_{l=1}^{w} s_{i,l} s_l = s_{i,l'} s_{l'} \neq s_{l'} \bmod N$$

and the probability that

$$D(C_i) = D(\prod_{l=1}^{w} c_{i,l}^{s_l}) = \sum_{l=1}^{w} s_{i,l} s_l = s_{i,l'} s_{l'} = s_{l''} \bmod N$$
$$l'' \neq l'$$
$$1 \leq l'' \leq w$$

is $(w-1)/K$ as $s_1, s_2, \ldots, s_w$ are randomly chosen in Z_K. Therefore, the probability that

$$D(C_i) = s_{l''} \bmod N$$
$$1 \leq l'' \leq w$$

is negligible.

In the second case, suppose only $s_{i,T_1}, s_{i,T_2}, \ldots, s_{i,T_\pi}$ are non-zero integers in $s_{i,1}, s_{i,2}, \ldots, s_{i,w}$ where $1 \leq T_1, T_2, \ldots, T_\pi \leq w$ and $\pi > 1$. Then

$$D(C_i) = D(\prod_{l=1}^{w} c_{i,l}^{s_l}) = \sum_{l=1}^{w} s_{i,l} s_l = \sum_{l=1}^{\pi} s_{i,T_l} s_{T_l} \bmod N.$$

So, as $s_1, s_2, \ldots, s_w$ are randomly chosen in Z_K the probability that

$$D(C_i) = s_{l''} \bmod N$$
$$1 \le l'' \le w$$

is *w/K* and thus negligible.

Therefore, in both cases the probability that the message encrypted in C_i lies in *S* is negligible.

Proof that $C_i = \prod_{l=1}^{w} c_{i,l}^{s_l} \bmod N^2$ encrypts a message in *S* can be implemented by V_i using the following proof

$$KN[\log_{R_i^N}(C_i/g^{s_1})] \vee KN[\log_{R_i^N}(C_i/g^{s_2})] \vee KN[\log_{R_i^N}(C_i/g^{s_w})] \quad (3)$$

where R_i can be published. It is a proof of knowledge of 1-out-of-*w* discrete logarithms.Although ZK proof of partial knowledge [4] can implement it, the naive solution costs *O(w)* exponentiations in both proof and verification and is inefficient as explained in Section 2.

3.2 Efficient Proof with an Honest Verifier

The main idea of our efficient vote validity proof is proposed first in a prototype, which employs an honest verifier for simplicity of description. The talliers set up w different Diffie-Hellman keys, each of whose discrete logarithm is the product of two key roots. The two key roots for the j^{th} Diffie-Hellman key include a constant key root chosen by the talliers and $\log_{R_i^N} y_j$ where $y_j = C_i/g^{s_j} \bmod N^2$. The talliers then seal a random secret message *m* into *w* commitments, using a hash function *H( )* and the *w* Diffie-Hellman keys. The talliers publish the *w* commitments and his Diffie-Hellman commitment of his key root and then asks the voter V_i to extract *m*. Obviously, V_i can extract *m* if he knows one of the *n* Diffie-Hellman keys, while Diffie-Hellman assumption (to be detailed later) implies that he knows the j^{th} Diffie-Hellman key if and only if he knows $\log_{R_i^N} y_j$. The proof protocol is described in Fig 2, which guarantees that the voter knows one of $\log_{R_i^N} y_1, \log_{R_i^N} y_2, \ldots, \log_{R_i^N} y_w$ under Diffie-Hellman assumption.

This new vote validity proof protocol is called, "Protocol 1," in this paper. It is quite efficient. Especially the prover only needs a low constant computational independent of *w* in Protocol 1, which is thus suitable for low-capability voters who using mobile devices. Moreover, as *L*<<*N*, it greatly improves communicational efficiency as well. Completeness of Protocol 1 is straightforward and any interested reader can follow the proof protocol step by step to check it. Its soundness is proved in Theorem 2 under Diffie-Hellman assumption, as defined in Definition 5, while it is proved to be honest-verifier zero knowledge in Theorem 3.

Definition 5: (Diffie-Hellman assumption) Given $h^a \bmod N^2$ and $h^b \bmod N^2$, it is hard to calculate $h^{ab} \bmod N^2$ unless either *a* or *b* is known.

Theorem 2: Passing Protocol 1 guarantees the voter's knowledge of one of $\log_{R_i^N} y_1, \log_{R_i^N} y_2, \ldots, \log_{R_i^N} y_w$ under Diffie-Hellman assumption.

Proof: Passing Protocol 1 implies

$$u = m$$

For simplicity and without losing generality, suppose V_i knows $x_t = \log_{R_i^N} y_t$ where $1 \le t \le w$. A one-way and collision-resistent hash function $H()$ from Z_{N^2} to Z_L is employed where $L << N$.

1. The talliers randomly choose an integers r from Z_N and a random message m from Z_L. They calculate and conceal

$$z_j = y_j^r \bmod N^2 \text{ for } j = 1, 2, \ldots, w.$$

They calculate and publish

$$c_j = H(z_j) \oplus m \text{ for } j = 1, 2, \ldots, w$$
$$z = (R_i^N)^r \bmod N^2.$$

2. V_i publishes

$$u = H(z_t) \oplus c_t$$

where $z_t = z^{x_t} \bmod N^2$.

3. The talliers verify

$$u = m.$$

They accept the proof iff this equation holds.

Fig. 2. Efficient proof of vote validity with an honest verifier

and thus the user knows *m*. As the only published information about *m* is

$$c_j = H(z_j) \oplus m \ \ for \ \ j = 1,2,\ldots,w,$$

the user must know a z_J where $1 \le J \le w$. So according to Diffie-Hellman assumption, the user must know $x_J = \log_{R_i^N} y_J$ as

$$z_J = (R_i^N)^{x_J r} \bmod N^2$$

and *r* is kept secret from the prover.

Theorem 3: Protocol 1 achieves honest-verifier zero knowledge.

Proof: The proof transcript of Protocol 1 contains $c_1, c_2, \ldots, c_w, z, u$. A party without any knowledge of any secret can simulate the proof transcript as follows.

1. He randomly chooses an integer *r* from Z_N and a random message *m* from Z_L.
2. He calculates $z = (R_i^N)^r \bmod N^2$ and $z_j = y_j^r \bmod N^2$ *for* $j = 1,2,\ldots,w$.
3. He calculates $c_j = H(z_j) \oplus m$ *for* $j = 1,2,\ldots,w$
4. He sets *u*=*m*.

In both the real transcript of Protocol 1 with an honest verifier and the simulating transcript,

- *u* is uniformly distributed in Z_L.
- *z* is uniformly distributed in the cyclic group generated from the generator R_i^N.
- $c_j = H(z_j) \oplus u$ for *j=1,2,...w* where $z_j = y_j^{\log_{R_i^N} z} \bmod N^2$.

As the two transcripts have just the same distribution, if the verifier is honest, a party without any knowledge of any secret and that does not deviate from the proof protocol can simulate the proof transcript of Protocol 1 without any difference. So Protocol 1 achieves honest-verifier zero knowledge.

Note that proof of zero knowledge of Protocol 1 in Theorem 3 assumes that the verifier is honest and does not deviate from the proof protocol. As explained in Section 2.1, it is very common and actually necessary in many proof protocols. However, when analysing privacy of a proof protocol, the assumption is not always unconditionally necessary and inherent. A verifier may be dishonest and deviate from a proof protocol in order to obtain some secret information

from the prover. So if privacy of a proof protocol can only be proved when assuming that the verifier is honest, it should be explicitly called an honest-verifier zero knowledge proof protocol. Moreover, if the assumption can be removed in a proof protocol, it can achieve stronger and more formal privacy.

3.3 Efficient Vote Validity Proof with a Dishonest Verifier

A dishonest verifier in Protocol 1 can deviate from the protocol and try to extract some information about t as illustrated in Fig 3. Simply speaking, it commits to w different messages in $c_1, c_2, \ldots, c_w$ respectively and can determine t according to the message returned by the user.

Using this attack, no matter how t distributes in $\{1,2,\ldots,w\}$ a dishonest verifier can always extract t. A simple counteract to this attack is to slightly modify Protocol 1 into the proof protocol

1. A verifier randomly chooses an integers r from Z_N and w different message $m_1, m_2, \ldots, m_w$ from Z_L. He calculates and conceals

$$z_j = y_j^r \bmod N^2 \text{ for } j = 1, 2, \ldots, w.$$

The verifier calculates and publishes

$$c_j = H(z_j) \oplus m_j \text{ for } j = 1, 2, \ldots, w$$
$$z = (R_i^N)^r \bmod N^2.$$

2. A_t publishes

$$u = H(z_t) \oplus c_t$$

where $z_t = z^{x_t} \bmod N^2$.

3. The verifier obtains

$$u = m_t$$

and thus t.

Fig. 3. Attack by a dishonest verifier

Suppose V_i knows $x_t = \log_{R_i^N} y_t$ where $1 \le t \le w$.

1. A verifier randomly chooses an integers r from Z_N and a random message m from Z_L. He calculates and conceals

$$z_j = y_j^r \bmod N^2 \text{ for } j = 1, 2, \ldots, w.$$

The verifier calculates and publishes

$$c_j = H(z_j) \oplus m \text{ for } j = 1, 2, \ldots, w$$
$$z = (R_i^N)^r \bmod N^2.$$
$$M = H'(m||m||\ldots||m)$$

where $H'()$ is a one way and collision-resistent hash function from Z_{L^K} to Z_L and m is concatenated to itself repeatedly for k times before being input to $H'()$.

2. A_t calculates

$$u = H(z_t) \oplus c_t$$

where $z_t = z^{x_t} \bmod N^2$. He tests whether $H'(u||u||\ldots||u) = M$. If it holds, he publishes u; otherwise he gives up his proof as the verifier is detected to be dishonest.

3. If A_t publishes u, the verifier verifies

$$u = m.$$

He accepts the proof iff the equation holds.

Fig. 4. Efficient vote validity proof with a dishonest verifier

in Fig 4, which is called Protocol 2. In Protocol 2, the verifier must commit to m in a separate commitment using a one-way and collision-resistent hash function when generating $c_1, c_2, \ldots, c_w$. Before returning *u*, the prover tests whether the hash function of *u* equals the separate commitment. He publishes *u* if and only if the test is passed.

Completeness and soundness of Protocol 1 are maintained in Protocol 2 as the latter only employs an additional operation commitment of *m* by the verifier and verification of the commitment by the voter, which does not affect the voter's proof. Moreover, privacy of Protocol 1 is improved in Protocol 2. In Protocol 2, if a dishonest verifier launches the attack in Fig 3, he can only succeed with a probability no larger than *1/w*. More precisely, if $M = H'(m_j \| m_j \| \ldots \| m_j)$, he can only succeed when *t=j*. When $t \neq j$, the user will find $H'(u \| u \| \ldots \| u) \neq M$ and refuse to publish *u*. So Protocol 2 can reduce effectiveness of a dishonest verifier's attack to a low level.

3.4 Efficient Vote Validity Proof with an Untrusted Verifier

When the verifier cannot be trusted and attacks by him to extract *t* must be prevented, the vote validity proof protocol can be optimised into Protocol 3 as described in Fig 5. The idea of the optimisation is simple: before the voter returns *m* he needs to verify that all the $c_1, c_2, \ldots, c_w$ are encryptions of *m*.

As the employed hash functions are one-way and collision-resistent and it is difficult to find different integer pairs (k_1, m_1) and (k_2, m_2) in Z_L^2 such that $E_{k_1}(m_1) = E_{k_2}(m_2)$ and $E_{m_1}(k_1) = E_{m_2}(k_2)$, the verifier is guaranteed to have committed to the same *m* in the *w* instances of commitment (encryption). So it is not necessary to be trusted. Except that the user returns opening of one of $c_1, c_2, \ldots, c_n$ in two steps (instead of in one step) and there is a verification against the verifier before the second step, Protocol 3 actually employs the same proof principle as Protocol 1. So Protocol 3 inherits completeness and soundness of Protocol 1. Moreover, it optimises its zero knowledge property to achieve stronger privacy without any trust on the verifier.

Suppose V_i knows $x_t = \log_{R_i^N} y_t$ where $1 \leq t \leq w$. Besides one-way and collision-resistent hash functions $H()$ and $H'()$, a block cipher $E_k()$ (e.g. AES) with key k and a message space Z_L is employed.

1. The talliers randomly choose an integers r from Z_N and a random message m from Z_L. They calculate and conceal

$$z_j = y_j^r \bmod N^2 \text{ for } j = 1, 2, \ldots, w.$$

They calculate and publish

$$c_j = E_{H(z_j)}(m) \text{ for } j = 1, 2, \ldots, w$$
$$c'_j = E_m(H(z_j)) \text{ for } j = 1, 2, \ldots, w$$
$$z = (R_i^N)^r \bmod N^2.$$

2. V_i calculates

$$u = H(z_t) \oplus c_t$$

where $z_t = z^{x_t} \bmod N^2$ and verifies

$$E_{D_u(c'_j)}(u) = c_j \text{ for } j = 1, 2, \ldots, w$$

where $D()$ is the decryption function of $E()$. He publishes u if all the w equations hold. Otherwise he gives up his proof as the verifier is detected to be dishonest.

3. The talliers verify

$$u = m.$$

They accept the proof iff this equation holds.

Fig. 5. Efficient vote validity proof with an untrusted verifier

4. COMPARISON AND CONCLUSION

The new vote validity proof and verification protocols are compared with the traditional vote validity proof and verification technique in homomorphic e-voting through ZK proof of partial knowledge [4] in Table 1. In comparison of privacy, the ZK models in different solutions are listed. In comparison of computational efficiency, computational cost of a voter and a verifier (tallier) in proving and verifying validity of the vote is estimated and exponentiations with full-length exponents in Z_N are counted. The proof protocols in Fig 2, Fig 4, and Fig 5 are very efficient as the number of public key cryptographic operations (in the form of exponentiations with exponents in Z_N) is small. However, the three proof protocols need an additional cost: a verifier in them needs to calculate $C_i = \prod_{l=1}^{w} c_{i,l}^{s_l} \bmod N^2$, C_i/g^{s_1}, C_i/g^{s_2}, $\ldots C_i/g^{s_w}$. Those additional operations seem to cost *2w* exponentiations with full length exponents if *K* is as large as *N*. Fortunately, as *N* is very large (e.g., hundreds of bits long) *K* can be much smaller than *N* while *w/K* is still negligible to guarantee the validity of the votes. For example, *K* can be set as $N^{1/w}$ in most election applications. When *N* is 1024 bits long and *w=4*, *K* is 256 bits long and is thus large enough. When *N* is 1024 bits long and *w=16*, *K* is 64 bits long and *w/K* is still negligible. With this optimised setting, cost of the additional operations is approximately equal to 2 exponentiations with full-length exponents in Z_N and high efficiency is achieved for not only the prover but also for verifier. In comparison of communicational efficiency, the number of bits transferred between a prover (voter) and the verifier(s) is estimated. In a typical example where *w=8*, $\log_2 N = 1024$ and $\log_2 L = \log_2 K = 128$, efficiency advantage of our new methods in communication is significant as well.

The comparison clearly demonstrates that in comparison with the existing solution our new vote validity proof and verification technique is more efficient and stronger in privacy.

Table 1. Comparison of Vote Validity Check

Method	Computation		Communication	ZK
	voter	verifier		
[4]	4 *w*	4 *w*	$4w\log_2 N$	honest verifier
Protocol 1	1	*w* + 3	$w(\log_2 K + \log_2 L) + 2\log_2 N + \log_2 L$	honest verifier
Protocol 2	1	*w* + 3	$w(\log_2 K + \log_2 L) + 2\log_2 N + 2\log_2 L$	handling dishonest verifier
Protocol 3	1	*w* + 3	$w(\log_2 K + 2\log_2 L) + 2\log_2 N + \log_2 L$	no trust needed

REFERENCES

[1] James M. Adler and Wei Dai and Richard L. Green and C. Andrew Neff. "Computational Details of the VoteHere Homomorphic Election System". Technical Report, VoteHere Inc, 2000. Available from http://www.votehere.net/technicaldocs/hom.pdf, last accessed 22 June, 2002.

[2] Olivier Baudron and Pierre-Alain Fouque and David Pointcheval and Jacques Stern and Guillaume Poupard. "Practical multi-candidate election system". Twentieth Annual ACM Symposium on "Principles of Distributed Computing", 2001, pp.274-283.

[3] D Boneh and X Boyen. "Short signatures without random oracles". Eurocrypt '04 in Lecture Notes in Computer Science, pp.56-73.

[4] R Cramer and I Damgård and B Schoenmakers. "Proofs of partial knowledge and simplified design of witness hiding protocols". CRYPTO '94 in Lecture Notes in Computer Science, Berlin, 1994.

Springer-Verlag, pp.174-187.
[5] Ivan Damgaård and Mats Jurik. “A Generalisation”, A Simplification and Some Applications of Paillier's Probabilistic Public-Key System. Public Key Cryptography-PKC 01, 2001, pp.119-136.
[6] I Damgård. “Efficient Concurrent Zero-knowledge in the Auxiliary String Model”. EUROCRYPT '00 in Lecture Notes in Computer Science, 2000, pp.431-444.
[7] P Fouque and G Poupard and J Stern. “Sharing Decryption in the Context of Voting or Lotteries”. Financial Cryptography 2000, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science, pp.90-104.
[8] Pierre-Alain Fouque and Guillaume Poupard and Jacques Stern. “Sharing Decryption in the Context of Voting or Lotteries”. Financial Cryptography 2000, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science, pp.90-104.
[9] J Groth. “Non-interactive Zero-Knowledge Arguments for Voting”. ACNS '05, Berlin, 2005. Springer-Verlag. Lecture Notes in Computer Science, pp.467-482.
[10] Guillou, L. C. and Quisquater, J. J. “A paradoxical" identity-based signature scheme resulting from zero-knowledge”. In Shafi Goldwasser, editors, CRYPTO '88 in Lecture Notes in Computer Science, Berlin, 1989. Springer-Verlag, pp.216-231.
[11] Martin Hirt and Kazue Sako. “Efficient Receipt-Free Voting Based on Homomorphic Encryption”. Advances in Cryptology-EUROCRYPT 00, 2000, pp.539-556.
[12] Jonathan Katz and Steven Myers and Rafail Ostrovsky. “Cryptographic Counters and Applications to Electronic Voting”. Advances in Cryptology-EUROCRYPT 01, 2001, pp.78-92.
[13] Aggelos Kiayias and Moti Yung. “Self-tallying Elections and Perfect Ballot Secrecy”. Public Key Cryptography, 5th International Workshop-PKC 02, 2002, pp.141-158.
[14] Byoungcheon Lee and Kwangjo Kim. “Receipt-free Electronic Voting Scheme with a Tamper-Resistant Randomizer”. Information Security and Cryptology, ICISC 2002 in Lecture Notes in Computer Science, 2002. Springer-Verlag, pp.389-406.
[15] Byoungcheon Lee and Kwangjo Kim. “Receipt-Free Electronic Voting Through Collaboration of Voter and Honest Verifier”. JW-ISC 2000, 2000, pp.101-108.
[16] C. Andrew Neff. “Conducting a Universally Verifiable Electronic Election Using Homomorphic Encryption”.White Paper, VoteHere Inc, 2000.
[17] K Peng and F Bao. “Efficient multiplicative homomorphic e-voting”. ISC '10 in Lecture Notes in Computer Science, 2010, pp.381-393.
[18] K Peng and F Bao. “Efficient Proof Of Validity Of Votes In Homomorphic E-Voting”. NSS (International Conference on Network and System Security) '10, 2010, pp.17-23.
[19] Kun Peng and Feng Bao. “Efficient Vote Validity Check in Homomorphic Electronic Voting”. ICISC 2008 in Lecture Notes in Computer Science, 2008, pp.202-217.
[20] Kun Peng and Colin Boyd and Ed Dawson and Byoungcheon Lee. “Multiplicative Homomorphic E-Voting”. INDOCRYPT 2004 in Lecture Notes in Computer Science, Berlin, 2004. Springer-Verlag, pp.61-72.
[21] Berry Schoenmakers. “Fully Auditable Electronic Secret-Ballot Elections”. XOOTIC Magazine, 2000.

Kun Peng

He got his Bachelor degree in software engineering and his Master degree in computer security from Huazhong University of Science and Technology, China. He graduated from Information Security Institute, Queensland University of Technology, Australia in 2004, obtaining his PhD degree in information security. His main research interests include applied cryptology, network security, secure e-commerce and e-government. He is now a scientist at Institute for Infocomm Research, Singapore.