Homomorphic Voting Schemes

Panagiotis Grontas

$\mu\Pi\lambda\forall$ - CoReLab Crypto Group

22/11/2013

Introduction

- Voting secrecy $\rightarrow$ Encrypt the votes
- Calculate the result: Decrypt each vote and count
- What if we could calculate the result without decrypting the votes;
- Solution: Homomorphic Cryptosystems
  - Encrypt votes under authority public key
  - Combine ciphertexts
  - $E(v_1) \otimes \cdots \otimes E(v_n) = E(v_1 \oplus \cdots \oplus v_n)$
  - Result: Encrypted vote aggregate
  - Decrypt using the (shared) authority private key

Homomorphic Cryptosystems

Ballot secrecy under cryptographic assumptions

- The RSA and the El Gamal cryptosystem is multiplicatively homomorphic.
  - $E(m_1)E(m_2) = m_1^e m_2^e \pmod n = (m_1 m_2)^e \pmod n = E(m_1 m_2)$
  - $E(m_1)E(m_2) = (g^{r_1}, m_1 h^{r_1})(g^{r_2}, m_2 h^{r_2}) = (g^{r_1+r_2}, m_1 m_2 h^{r_1+r_2}) = E(m_1 m_2)$
- The exponential El Gamal cryptosystem is additively homomorphic.
  - $E(m_1)E(m_2) = (g^{r_1}, g^{m_1} h^{r_1})(g^{r_2}, g^{m_2} h^{r_2}) = (g^{r_1+r_2}, g^{m_1+m_2} h^{r_1+r_2}) = E(m_1 + m_2)$
- The Goldwasser Micali and the Benaloh Cryptosystems are additively homomorphic.
  - $E(m_1)E(m_2) = y^{m_1} r_1^2 y^{m_2} r_2^2 = y^{m_1+m_2}(r_1 r_2)^2 = E(m_1 + m_2)$
  - $E(m_1)E(m_2) = y^{m_1} x_1^r y^{m_2} x_2^r = y^{m_1+m_2}(x_1 x_2)^r = E(m_1 + m_2)$
- The Paillier cryptosystem is additively homomorphic.
  - $E(m_1)E(m_2) = (1+n)^{m_1} r_1^{n^s} (1+n)^{m_2} r_2^{n^s} \pmod{n^{s+1}} = E(m_1 + m_2)$

The homomorphic secret sharing approach

Initial Aim: Ballot secrecy under no assumptions

- A *super secret* S can be computed from m *sub secrets* $\{S_i\}_{i=1}^{m}$
  $S = f(S_1, ..., S_m)$
- Each of the sub secret holder acts as the dealer in the secret sharing scheme and deals the shares into n entities $\{E_j\}_{j=1}^{n}$ each receiving $\{S_{ij}\}_{i=1,j=1}^{m,n}$.
- Each entity combines its shares $R_j = g(S_{1j}, ..., S_{mj})$
- A subset of the entities reconstruct S using the secret sharing scheme.

$$S \leftarrow_f \begin{bmatrix} S_1 \\ S_2 \\ \cdots \\ S_m \end{bmatrix} \rightarrow_{share} \begin{bmatrix} E_1 & \cdots & E_n \end{bmatrix} \rightarrow \begin{bmatrix} S_{11} & \cdots & S_{1n} \\ S_{21} & \cdots & S_{2n} \\ & \cdots & \\ S_{m1} & \cdots & S_{mn} \end{bmatrix} \rightarrow \begin{bmatrix} g(S_{11}, \cdots, S_{1n}) \\ g(S_{21}, \cdots, S_{2n}) \\ \cdots \\ g(S_{m1}, \cdots, S_{mn}) \end{bmatrix} \rightarrow S$$

The problem

- Encryption promotes secrecy but hinders integrity, fairness
- Attacks:
  - Vote cancelling: Instead of encrypting a normal vote $v \in \{0, 1\}$, encrypt 1000 or -1000
  - Authority aggregates extra votes or discards votes
  - Refuse to decrypt the tally
- Solution: Add controls that increase
  - Fairness
  - Verifiability
  - Robustness

Benaloh Cryptosystem I

$r - residues$

y is an $r - residue \pmod n$ if $\exists x : y = x^r \pmod n$.
Trapdoor: n has known factorisation, it is easy to recognise an $r - residue$.

- **Key Generation**
  - Agree on a prime number r known to every participant.
  - Select randomly and independently two large primes p, q such that $r \mid (p-1)$ and $r \nmid (q-1)$
  - Calculate the RSA modulus $n = p \cdot q$
  - Select a *quadratic* ***non****-residue*. $y \in \mathbb{Z}_n^*$ with $gcd(y, n) = 1$.
  - The public key is N, y
  - The private key is p, q
- **Encryption**
  - $c = E(m) = y^m x^r \pmod n$ where x is random.

Benaloh Cryptosystem II

- **Decryption**
  - Factorisation of n is known
  - Calculate $\phi(n) = (p-1)(q-1)$
  - Calculate

$$u = E(m)^{\frac{\phi(n)}{r}} = (y^m x^r)^{\frac{\phi(n)}{r}} = y^{m\frac{\phi(n)}{r}} x^{\phi(n)} = y^{m\frac{\phi(n)}{r}}$$

  - If $u = 1$ then $m = 0$
  - Else iterate over all the possible $t \in \{0, \cdots, r-1\}$ checking if

$$uE(t) = E(m)E(t) = E(m+t) = E(0) = 1$$

  - Then $m = -t \pmod{n}$
  - Improve by baby step - giant step algorithm in $O(\sqrt{r})$ steps

Early Solutions: Benaloh-Fisher (1985) [CF85] I

- Ballot: encryptions of a yes-vote and a no-vote in random order $(yf^r \pmod n, g^r \pmod n)$
- Voter: generates $\eta + 1$ ballots
  - Master Ballot to be used $(\hat{yf}^r \pmod n, \hat{g}^r \pmod n)$
  - η test ballots to prove validity
- Beacon: source of randomness
  - Generate η random bits
  - If $b = 1$ reveal test ballot
  - If $b = 0$ reveal $\frac{f}{\hat{f}}, \frac{g}{\hat{g}}$
- Select one option as the vote :$v = \hat{yf}^r \pmod n$ or $v = \hat{g}^r \pmod n)$

Early Solutions: Benaloh-Fisher (1985) [CF85] II

- Tallying: Multiply the votes $\prod_{i=1}^{N} v_i = y^t x^r \quad (mod\, n)$
- Decrypt to calculate t
- Calculate x using brute force
- Prove tally correctness
  - The tallier generates η values c_i coprime to n and publishes $C_i = c_i^r$
  - The beacon generates η bits.
  - For all 1 beacon bits the tallier reveals c_i and for all 0 bits he reveals $c_i' = c_i x$.
  - The potential verifiers check that for the tally released: $y^t {c'}_i^r = C_i \prod_{i=1}^{N} v_i$

Early Solutions: Benaloh-Fisher (1985) [CF85] III

Properties

- Robust against voters
- Verifiable with probability $1 - 2^{-\eta}$
- Privacy against other voters
- Privacy against government?
- The government can progressively calculate the tally thus breaking privacy

Solution: Distribute the power of the tallier

Benaloh - Yung (1986) [BY86]

- The tallying function is distributed among k tallying authorities
- Each tallier multiplies its shares and retrieves its subtotal
- All the subtotals are added

Problem: Huge complexity $O(\eta N k)$ from:

- Ballot Validation
- Vote sharing

Reduction of complexity:
Witness Indistinguishable and Witness Hiding Proofs (Cramer, Damgard, Schoenmakers 1994 [CDS94])

Witness Indistinguishable Proofs I

- Relax the requirement of no information leakage for ZK Proofs
- The verifier should not be able to distinguish between equivalent witnesses eligible for the proof or
- The verifier might learn part of the witness and not the witness as a whole
- Framework to convert any three round honest verifier SK proof to WID

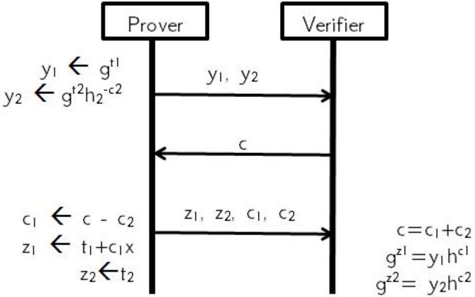
Witness Indistinguishable Proofs II

- Let $W = \{w_1, \cdots, w_n\}$ the set of alternative witnesses.
- For the actual witness used the prover calculates the offer dictated by the ZK protocol.
- For the alternate witnesses the prover calls the simulator, which returns the relevant offers that would cause the verifier to accept in a simulated transcript.
- The prover sends all the offers computed in the previous steps to the verifier.
- The verifier sends a random challenge.
- The prover interprets the challenge as a secret to be shared. The shares of the secret will be the random values employed by the simulator.
- The prover calculates the rest of the shares and the appropriate responses.
- The verifier validates the responses.

Witness Indistinguishable Proofs III

Example: Schnorr WID version

Prove knowledge of $x_1 : h_1 = g^{x_1}$ or $x_2 : h_2 = g^{x_2}$ without revealing which one.



Witness Indistinguishable Proofs IV

- **Offer**
  - For the actual witness x_1 calculate $y_1 = g^{t_1}$ where $t_1 \in_R \mathbb{Z}_q$
  - For x_2 the prover calculates $y_2 = g^{t_2} h_2^{-c_2}$ where $t_2 \in_R \mathbb{Z}_q$ and $c_2 \in_R \mathbb{Z}_q$ (interpret as random share of the challenge).
  - Send y_1, y_2 to the verifier.
- **Challenge**
  - The verifier challenges with the secret to be shared $c \in_R \mathbb{Z}_q$
- **Response**
  - The prover calculates the other part of the share $c_1 = c - c_2$
  - Actual witness response $z_1 = t_1 + c_1 x$
  - Simulated witness response $z_2 = t_2$.
  - Send responses z_1, z_2 and c_1, c_2 to the verifier.
- **Verification**
  - The verifier checks that the challenge was shared correctly $c = c_1 + c_2$
  - The verifier checks the offers conform the ZK protocol i.e. if $g^{z_1} = y_1 h_1^{c_1}$ and $g^{z_2} = y_2 h_2^{c_2}$

Cramer-Franklin-Schoenmakers-Yung ([CFSY96]) - 1996 I

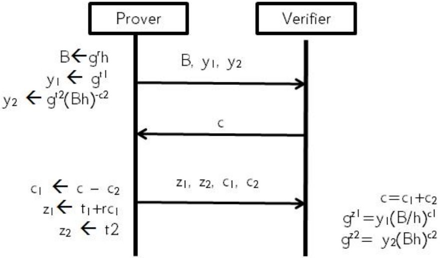
Features

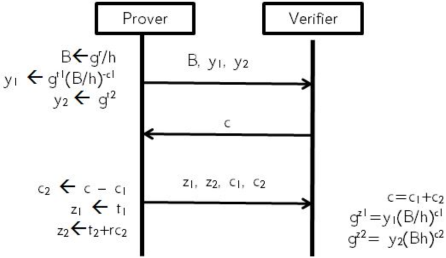
- Linear Complexity For Voter and Authority (n voters, k authorities)
- Pedersen Commitments and Verifiable Secret Sharing
- Private communication channels between voting and authorities
- Bulletin Board
- Masked voting enables the preference selection to be delayed

- **Ballot Construction** Each voter v_i:
  - Selects a masked vote b_i as a random value in $\{-1, 1\}$
  - Commits to b_i $B_i = g^{r_i} h^{b_i}$ where g, h, r_i are random

Cramer-Franklin-Schoenmakers-Yung ([CFSY96]) - 1996 II

- Proves that $b_i = 1$ or $b_i = -1$ by using a WID proof (a variation of the schnorr proof)





Cramer-Franklin-Schoenmakers-Yung ([CFSY96]) - 1996 III

- Shares r_i, b_i by computing $t-1$ degree polynomials $G_i(x), H_i(x)$ and commits to the coefficients $\{B_{il} = g^{r_{il}} h^{b_{il}}\}_{l=1}^{t-1}$
- The voter posts (B_i, validity proof, coefficient commitments) to the bulletin board, making them available to everybody
- The shares of (r_i, b_i) $\{r_{ij} = G_i(j), b_{ij} = H_i(j)\}_{j=1}^{k}$ are sent to the k authorities for validation encrypted using their public keys
- Attention: This opens up room for *didn't send/didn't receive* disputes since the BB is not used
- Each authority validates the shares they received against information from the BB
    - $g^{r_{ij}} h^{b_{ij}} = B_i \prod_{l=1}^{t-1} B_{il}^{j^l}$ where $B_i, \{B_{il}\}_{l=1}^{t-1}$
    - If everything is ok it holds since: $B_i \prod_{l=1}^{t-1} B_{il}^{j^l} = g^{G_i(j)} h^{H_i(j)}$.

Cramer-Franklin-Schoenmakers-Yung ([CFSY96]) - 1996 IV

- **Vote Casting**
  - b_i is not the final vote.
  - Select s_i such that $v_i \in \{-1, 1\}$ and $v_i = s_i b_i$.
  - s_i is posted to the BB
- **Tallying** using secret sharing homomorphisms
  - Authority A_j sums the shares r_{ij}, b_{ij} multiplied by voters' selected values s_i.
  - Authority A_j posts $S_j = \sum_{i=1}^{N} r_{ij} s_j$ and $T_j = \sum_{i=1}^{N} b_{ij} s_j$
  - Validation: A_j checks if $g^{S_j} h^{T_j} = \prod_{i=1}^{N} (B_i \prod_{l=1}^{t-1} B_{il}^{j^l})^{s_i}$
  - Final tally: $T = \sum_{j \in A} T_j \prod_{l \in A - \{j\}} \frac{l}{l-j}$

Cramer-Gennaro-Schoenmakers ([CGS97]) - 1997 I

Features

- Vote and go
- Optimal with respect to the voter (independent of the number of authorities)
- Linear work for the authorities wrt the voters
- Exponential El Gamal encryption for each ballot
- Threshold cryptosystem instead of secret sharing

Threshold El Gamal

- **Key Generation - VSS**
  - Each authority i chooses $x_i \in_R \mathbb{Z}_q$ to be his share of the key.
  - $f_i(z) = \sum_{j=0}^{t-1} f_{ij} z^j$ where $f_{i0} = x_i$ and $f_{ij} \in_R \mathbb{Z}_q$.
  - Commit to the coefficients $F_{ij} = g^{f_{ij}}$.
  - Send shares $s_i j = f_i(j)$ to participant j.
  - Each participant verifies the shares he received against the broadcasted ones, by checking if $g^{s_{ij}} = \prod_{l=0}^{t-1} F_{jl}^{i^l}$
  - Each authority i commits to the shares by announcing $y_i = g^{x_i}$
- **Encryption** The public key can be computed as $y = \prod_{i=1}^{k} y_i^{\lambda_i}$, where λ_i are Lagrange coefficients. Encryption can proceed as in regular El Gamal.
- **Combination** and **Decryption** of (G, M)
  - Each auhority calculates $w_i = G^{x_i}$.
  - The plaintext can be uncovered as: $\frac{M}{\prod_{i \in \Lambda} w_j^{\lambda_i}}$

  Required proof that $x_i = log_G w_i = log_g y_i$ (Chaum-Pedersen protocol).

CGS I

- **Ballot Construction**
  - A yes vote will be represented as $m_y = 1$ and a no vote as $m_n = -1$
  - Select a random $b \in \{1, -1\}$
  - Prepares the encryption $(x, y) = (g^r, h^r G^b)$
  - Prove Validity: $b = 1$ or $b = -1$
  - $log_g x = log_h y / G$ for $b = 1$ or $log_g x = log_h y G$ for $b = -1$

CGS II

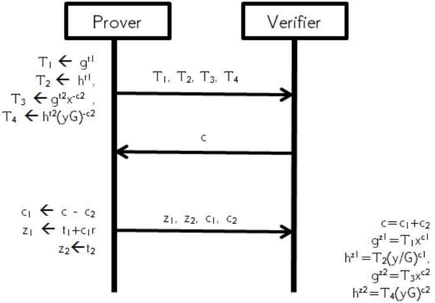


Figure : Proof of validity for yes ballot

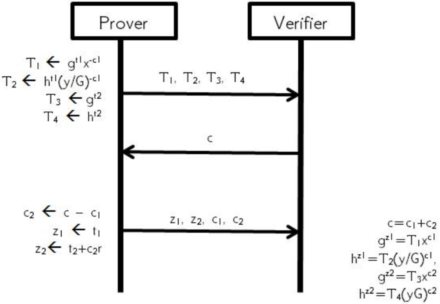


Figure : Proof of validity for no ballot

CGS III

- **Vote Casting** $v_i \in \{-1, 1\}$ by selecting s_i such that $v_i = s_i b_i$
- **Tallying**
  - All the valid votes are multiplied $(A, B) = (\prod_{i=1}^{N} g^r, \prod_{i=1}^{N} h^r G^v)$.
  - Share combination and decryption for threshold El Gamal
  - $W = \frac{A}{B^x} = G^{\sum_{i=1}^{N} v_i} = G^T$
  - $T = log_G W$ a discrete logarithm
  - Brute Force :$-N \leq T \leq N \rightarrow G, G^2, G^3, \cdots$

Extensions to multiple candidates I

- $C > 2$ candidates
- Option 1:$1 - out - of - C$ candidates
- Option 2:$t - out - of - C$ candidates
- A simple solution:
  - A super ballot for C yes-no elections
  - $1 - out - of - C$ elections 1 *yes* vote and $C - 1$ *no* votes
  - $t - out - of - C$ elections t *yes* vote and $C - 1$ *no* votes
  - C counters where ballots are aggregated

Extensions to multiple candidates II

- Application to CGS: C discrete logarithms - $G_1^{T_1} \cdots G_C^{T_C}$
- A problem: Discrete logarithm of a larger number
- A solution - Baudron Counters [BFP+01]
  - Select a number D and used as a *single counter*
  - Vote for candidate $c \rightarrow$ encrypt D^c.
  - Multiply the the encrypted votes
  - Tally to be decrypted $T = \sum_{c=0}^{C} t_c D^c \pmod{n^s}$, a number in $base - D$.
  - Decrypt T and retrieve the digits

Publicly Verifiable Secret Sharing

Main Idea

The validity of the dealer shares can be checked by everybody, and not only the participants

Motivation and Rationale

- Shamir secret sharing is fine as long as everybody is honest
- In reality, reconstruction might be obstructed because:
  - A corrupted dealer might send improper shares
  - The players replace their proper shares with invalid ones
  - Conflict resolution
- Solutions
  - Verifiable secret sharing against a corrupted dealer
  - Share publication against corrupted players
- An extension: proof of correctness for everything

Publicly Verifiable Secret Sharing - Process

Sharing

- Players: Register Public Keys
- Dealer: Polynomial Generation
- Publish Commitments to coefficients
- Calculate shares
- Encrypt shares using player public key
- Proof of correct encryption

Reconstruction

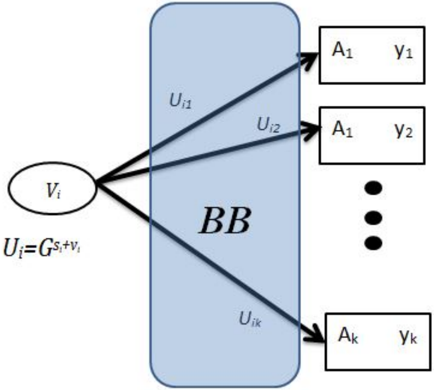
- Decrypt shares
- Proof of correct decryption
- Combine the shares and retrieve the secret

[Sch99] Security based on DLOG

Voting with Publicly Verifiably Secret Sharing Scheme I

- Bridge the gap between [CFSY96] and [CGS97]
- Secret sharing through the bulletin board
- Verifiable by everybody without disputes
- Properties:
    - **Voter**: Dealer
    - **Authority**: Player
    - **Secrecy**: Vote sharing
    - **Cheating voter:** Verifiable secret sharing
    - **Cheating authorities:** Publicly verifiable secret sharing

Voting with Publicly Verifiably Secret Sharing Scheme II



Discussion

Threshold cryptosystems vs PVSS in voting

- **Threshold scheme is executed between talliers**
- Large number of voters - small number of talliers
- Same group of talliers
- **PVSS - no interaction everything happens in the BB**
- Small scale elections
- Changing talliers
- Voter can be a tallier

The Paillier Cryptosystem

Main idea

$\mathbb{Z}^*_{n^2}$ is isomorphic to $\mathbb{Z}_n \times \mathbb{Z}^*_n$
$\mathbb{Z}^*_{n^s}$ is isomorphic to $\mathbb{Z}_n \times \mathbb{Z}^*_{n^{s-1}}$ where $n = pq$ and p, q are primes of the same length

Encrypt

- $E(m, r) = (1 + n)^m r^n \quad (mod\, n^2)$
- $E(m, r) = (1 + n)^m r^{n^{s-1}} \quad (mod\, n^s)$
- $(m, r) \in \mathbb{Z}_n \times \mathbb{Z}^*_n$

Decrypt

- Trapdoor $\phi(n)$ easy to compute given p, q
- $c' = c^{\phi(n)}$
- $m' = (c - 1)\, div\, n$
- $m = m'\, div\, \phi(n)$

CGS with Paillier I

CGS is a generic voting protocol so we can plug in Paillier instead of exponential El Gamal

- The voters encrypt their votes and prove that the encryption is either a yes/no vote. $E_i = E(v_i, r_i) = (1+n)^{v_i} r_i^{n^s} \pmod{n^{s+1}}$ where $v_i \in \{0, 1\}$
- The ciphertexts are posted on a bulletin board
- The talliers aggregate the votes, utilising the homomorhism
- $n^s < N^C$ which can be adjusted using the parameters n, s
- The result is placed in the BB
- A valid subset jointly decrypts the aggregate, and gets the result
- If instead of exponential El Gamal, Paillier is used, no need to brute force a search for DLOG
- The missing pieces:
  - prove whether an encryption corresponds to a yes or no vote
  - threshold version of Paillier: distributed exponentiation

Schnorr in Paillier I

- Prove that ciphertext c is a Paillier encryption of message m.
- Note that if $c = E(m, r) = (1+n)^m r^{n^s} \pmod{n^{s+1}}$
- Then $c(1+n)^{-m} = r^{n^s} \pmod{n^{s+1}}$.
- Equivalently $u = c(1+n)^{-m}$ is an encryption of zero
- Equivalently prove knowledge of randomness r such that u is a n^s power.

Schnorr in Paillier II

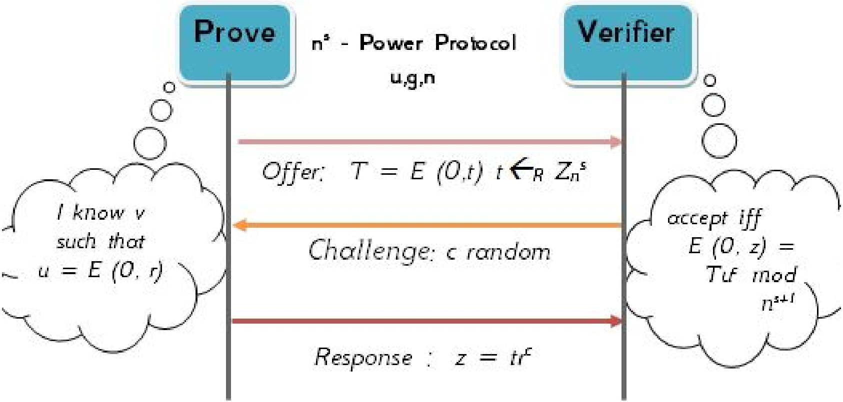


Figure : n^s Power Protocol - Proof of Knowledge of Randomness

Convert to WID proof

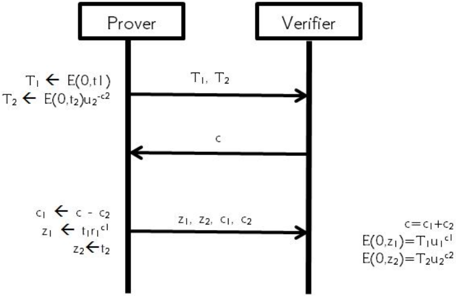


Figure : Witness indistinguishable proof of n^s power [DJN03]

Completeness: $E(0, z) = z^{n^s} = t^{n^s} r^{cn^s} = Tr^{n^{s^c}} = TE(0, r)^c = Tu^c$.

Multiple candidates with super ballot

- C parallel yes no votes $v_{ij}{}_{i=1,j=1}^{N,C}$
- Proof of validity: $E(v_{ij}, r_{ij})$ is n^s power
- Proof that exactly c candidates have been voted
    - Release product of randomness $\prod_{i=1}^{C} r_{ij}$
    - Calculate product of voter votes $V = \prod_{i=1}^{C} E(v_{ij}, r_{ij})$
    - Prove that $\frac{V}{(1+n)^c}$ is n^s power

Baudron counters I

- Vote for candidate c - Encrypt D^c
- $\sum_{j=0}^{C-1} a_j D^j < n^s \rightarrow ClogN < slogn$
- Prove vote validity: The vote indeed encrypts D^c where $c \in \{0, ..., C-1\} \Leftrightarrow c < C$

Strategy

- Commit to all η bits of the witness
- Prove that each commitment corresponds to $0, 1$
- Prove that the commitments do not leave out any bits

Conclusion: Since k commitments are used the witness lies in $[0, 2^{k+1} - 1]$

Baudron counters II

Helper: A protocol to prove that encryptions e_a, e_b, e_c correspond to plaintext a, b, c such that $ab = c \pmod{n^s}$

Baudron counters III

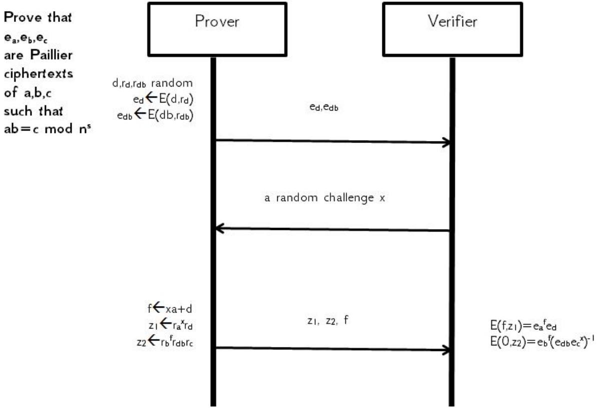


Figure : Proof of Paillier encrypted product [DJN03]

Baudron counters IV

- **Application Case 1:** The number of candidates C is a power of 2 $C = 2^{l+1}$ All binary numbers with $l+1$ bits are valid candidates
  - Convert the candidate index to its binary representation
  - $c = b_0 2^0 + b_1 2^1 + \cdots + b_l 2^l$ where $b_j \in 0, 1$. Then $D^c = D^{2^{0}{}^{b_0}} D^{2^{1}{}^{b_1}} \cdots D^{2^{l}{}^{b_l}}$
  - D^c is a product of powers of two or ones.
  - The voter encrypts each product term providing encryptions $e_0 = D^{2^{0}{}^{b}_{0}}, e_1 = D^{2^{1}{}^{b}_{1}}, \cdots, e_l = D^{2^{l}{}^{b}_{l}}$
  - Use a WID protocol to prove that $\frac{e_i}{(1+n)^1}$ or $\frac{e_i}{(1+n)^{D^{2^i}}}$ is an n^s power
  - Calculates the partial products $E_i = \prod_{x=1}^{i} D^{2^{x}{}^{b}_{x}}$ such that $E_l = D^c$
  - Use the product proof with $a = E_{i-1}, b = e_i, c = E_i$

Baudron counters V

- **Application Case 2:** The number of candidates C is not a power of 2 Some binary numbers with $l+1$ bits are valid candidates (the ones less than C)
  - Follow the same 3 steps as case 1
  - Define $\beta_i = (D^{2^i} - 1)^{-1} \pmod{n^s}$
  - Compute $e'_i = \frac{e_i}{(1+n)}^{\beta_i} \pmod{n^2} = Enc(b_i)$
  - $C = (B_l \cdots B_0)_2$
  - $c < C \Leftrightarrow \exists i > 0 : B_i = 1$ and $b_i = 0$ and $\forall j > i : B_j = b_j$
  - Need to prove equality of j indices and inequality of i
  - $B_i = b_i \Leftrightarrow z_i = \frac{(2B_i-1)(2b_i-1)+1}{2} = 1$
  - $Z_i = z_l \cdots z_{i+1} B_i(b_i - 1) = 1 \Leftrightarrow c < C$
  - Use the product proof with $a = Z_{i+1}, b = z_i, c = Z_i$
  - Use WID of n^s power protocol to show that $\exists i : Z_i = 1$

References I

Josh Cohen Benaloh.

Secret sharing homomorphisms: Keeping shares of a secret sharing.

In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 1986.

Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard.

Practical multi-candidate election system.

In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, PODC '01, pages 274–283, New York, NY, USA, 2001. ACM.

References II

Josh C Benaloh and Moti Yung.
Distributing the power of a government to enhance the privacy of voters.
In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, PODC '86, pages 52–62, New York, NY, USA, 1986. ACM.

Ronald Cramer, Ivan Damgård, and Berry Schoenmakers.
Proofs of partial knowledge and simplified design of witness hiding protocols.
In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 174–187, London, UK, UK, 1994. Springer-Verlag.

References III

Josh D. Cohen and Michael J. Fischer.

A robust and verifiable cryptographically secure election scheme (extended abstract).

In *FOCS*, pages 372–382, 1985.

Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung.

Multi-authority secret-ballot elections with linear work.

pages 72–83. Springer-Verlag, 1996.

Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers.

A secure and optimally efficient multi-authority election scheme.

pages 103–118. Springer-Verlag, 1997.

References IV

Ivan Damgård and Mats Jurik.

A generalisation, a simplification and some applications of paillier's probabilistic public-key system.

In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 119–136, London, UK, UK, 2001. Springer-Verlag.

Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen.

A generalization of paillier's public-key system with applications to electronic voting.

P Y A RYAN, page 3, 2003.

Jonathan Katz and Yehuda Lindell.

Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series).

Chapman & Hall/CRC, 2007.

References V

Berry Schoenmakers.

A simple publicly verifiable secret sharing scheme and its application to electronic voting.

In *In CRYPTO*, pages 148–164. Springer-Verlag, 1999.

Wikipedia.

Benaloh cryptosystem, 2012.

[Online; accessed 2-September-2013].