New cryptographic election protocol with best-known theoretical properties... and seems good enough for practical use

Warren D. Smith

Email: `warren.wds@gmail.com`;
URL: `math.temple.edu/~wds/homepage/works.html`
(paper=#89 there; see also the voting+crypto survey paper 80)

There are (now) two practical good-security approaches to running crypto-secure secret-ballot elections... BB-homo methods based on Bulletin Boards and homomorphic encryption, and our new method.

	BB-homo	New scheme
No double, fake, or invalid votes	✓	✓
Only listed authorized voters vote	✓	✓
Get correct election result	✓	✓
Provides NIZK proofs of above claims	✓	✓
Can prove did/didn't vote	✓	
"Coercion-resistant"		✓
Work (# modular expntns)	$O(N+V)$	$O(N+V)$
Storage (if V votes by N voters)	$O(N)$	$O(N+V)$
Communctn (above voting+BB downlds)	$O(1)$	$O(N+V)$
Heavily studied	✓	
Applicable if votes are...	additive	anonymizable
Uses homomorphic encryption/mixnets	Homo.E.	Mixnets

How BB-homo works (simplified):

0. Prepare list of authorized voters & keys to read their signatures.
1. Voters post signed encrypted vote, & ZK-validity proof, on BB.
2. Except: BB reencrypts vote before posting (provides voter with designated-verifier ZK-proof it validly did so).
3. Votes summed in encrypted form by multiplying encryptions.
4. Decryption key is immaculate shared secret. Sharers cooperatively decrypt total.

Encryption of vote V (g, h, t fixed public random elements in fixed public elliptic curve group of prime$\approx 2^{256}$ order; $t = g^{\ell}$; ℓ =immaculate shared secret; r =secret variable random integers):

$$V \to (g^r, h^V t^r)$$

.

How JCJ works (simplified):

0. Initially assume have pre-prepared list of authorized voters & their encrypted credentials. (No individual knows en/decryption transform; each voter's credential is private bitstring.)

1. Collect, mix & encrypt votes. (Votes include timestamps, credentials, & ZK-validity proofs; decryption key is immaculate shared secret but encryption key is public.)

2. By self-comparison of vote-list credentials via Plaintext Equality Testing (PET) remove all but one of each equivalence-class of identically-credentialed votes. [This one could (optionally) be the chronologically last.]

3. Remove timestamps from votes.

4. Re-mix & re-encrypt resulting pruned timestamp-free vote list.

5. Compare votes via PET to (mixed & encrypted) official credential list; remove bogus-credentialed votes. Mix+re-encrypt.

6. Cooperatively-decrypt, post, & then count (plaintext) votes.

Disadvantages of JCJ

1. Quadratic time - slow!:

- Self-comparison: $\binom{V}{2}$ comparisons via PET for V votes.
- Comparison of V votes to N-voter credential list: VN PETs.

2. In addition to being slow, the total communication of ZK-proofs that the PETs were done right, is uadratic in size. Unacceptable if $V \geq N \geq 10^8$.

3. Anything much slower than $O(V + N)$ permits "denial of service attacks" – submitting many bogus votes effectively cancels election.

Tools/ideas for fix:

1. Can we use *hashing* of credentials to perform self-comparison and official-list comparison tasks in only $O(V + N)$ steps?

2. "Secret encryptions" = no individual knows either the encryption or decryption process. (Some mutually distrustful people *cooperatively* know, but won't reveal since distrust.)

Development of idea

Yes, the 2 ideas can be made to work. First realized using Secure General MultiParty Computation (SGMPC) as "big gun." But SGMPC=extremely slow; resulting "$O(V+N)$" scheme slower than $O(V^2+VN)$ quadratic scheme if $V<10^9$. Impractical!

Next devised *special purpose* MPCs to speed up & simplify computation. Result: total work$\approx 100(V+N)$ exponentiations in an elliptic curve group and total communication $\approx 100(V+N)$ packets (each, say, 1Kbit) taking 30 hours to transmit even over single 1 GHz line if $V+N\approx 10^9$.

Dan Bernstein ECC exponentiation software: $1.4M$ pentium cycles max, so <0.35msec at 4GHz. Then 1000 computers (costing $\$10^6 \ll \$0.01$ per voter) do it in <10 hours even without special hardware. Now start long details detour...

ElGamal public-key Encryption & Decryption

Secret message M. Decryptor publishes two fixed random group elements g, h in elliptic curve group of prime order $P \approx 2^{256}$. Decryptor secretly knows ℓ where $h = g^\ell$. If elliptic curve discrete log problem is hard, infeasible for anybody else to determine ℓ.

Can encrypt M as 2-tuple $(g^r, h^r M)$ where r is a random nonzero integer mod P. (Note: because of r, encrypt same M twice $\Rightarrow$ different encryptions.)

Decryption: divide $h^r M$ by $(g^r)^\ell$ to get M.

Reencryption: $(g^r, h^r M) \to (g^{r+s}, h^{r+s} M)$.

ZK proofs of same exponent ($\mathrm{DL}^=$)

Peter Prover: knows two publicly known quantities $x = g^\ell$ and $y = h^\ell$ have *same* discrete logarithms ℓ (to public bases g and h).

Wishes to convince Vera Verifier of this – but without revealing what ℓ is. Procedure (Chaum&Pedersen, early 1990s):

1. Peter chooses random r mod P (but keeps it private);
2. Peter computes & prints $a = g^r$ & $b = h^r$;
3. Vera chooses random challenge c mod P & tells it to Peter;
4. Peter computes & prints $z = r + \ell c \bmod P$;
5. Vera verifies that $g^z = ax^c$ & $h^z = by^c$.

[And protocol can be made non-interactive (NI) by Fiat-Shamir hash trick: make challenge $c =$STANDARD-SECURE-HASH(x, g, y, h, a, b), which *Peter* computes & publishes, & Vera merely verifies.]

ZK proofs of: encryption, knowledge of plaintext, and exponentiation validity

1. Using above NIZK $\mathrm{DL}^=$ protocol, Peter can convince Vera that he's produced an ElGamal encryption $(g^\ell, h^\ell M)$ of a message M provided by Vera, but without revealing the secret key ℓ (the group elements g and h are public keys). Or he can show that $(g^{r+\ell}, h^{r+\ell} M)$ is an ElGamal reencryption of the original encryption, without revealing r.

2. Peter can prove knowledge of ℓ in the encryption $(g^\ell, h^\ell M)$, thus proving knowledge of the plaintext M, but again without revealing either ℓ or M.

3. Peter can compute X^z and prove he used the same z as for a previous Y^z.

ANDing and ORing (NI)ZK proofs

ZK-prove logical AND of two claims: simply present ZK-proofs of both. Indecomposable AND of two NIZK proofs involves "challenges" inside it that are constructed from a secure hash function of *both* component proofs. The point: some enemy cannot now surgically excise component NIZK proofs and glue them together with other components to get his own NIZK ANDed proof of something else – well, can, but the resulting proof will not have hashing property and hence obviously produced by surgery by somebody unauthorized, not by original authorized prover.

ZK-prove logical OR of two claims [not revealing which]:

$$\text{ZK-proof}_c(A \vee B) \equiv \text{ZK-proof}_d(A) \wedge \text{ZK-proof}_e(B) \wedge \{d \oplus e = c\}$$

where the subscripts c, d, e of the proofs denote the integer "challenges" presented to the prover by the verifier. (Prover can "forge" one proof...)

"Designated verifier" ZK proofs

...A brilliantly simple idea.

To ZK-prove statement X in such a way that only Bob will believe your proof:
ZK-prove: "X OR (proof of knowledge of Bob's secret key)."

Bob: "of course, this person does not know my secret key, so X must be true."

Alice: "Bob could have told this person his key (actually in typical use 'this person' *is* Bob). So I have no reason to believe X."

Note Bob cannot re-use the proof he is given to convince anybody else of X.

Bitstring "commitments"

Can commit n bits of information by publishing an AES-like encryption of a $(n+2s)$-bit message consisting of those n bits padded with s one-bits followed by s random bits (s is a security parameter). Can later reveal the committed bits by revealing (s-bit) secret encryption/decryption key.

(Other schemes also possible, e.g. commit x by publishing $g^x h^r$ where r random and g, h fixed public random group elements.)

Verifiable Shamir Secret sharing

Dealer who wants to share secret S selects random polynomial

$$F(x) = S + r_1 x + r_2 x^2 + \cdots + r_{t-1} x^{t-1}$$

of degree $< t$, & privately gives $S_j = F(j)$ to sharer j for $j = 1, 2, \ldots, Q$. Here S & the r_k are random integers mod P...

Verifiable Shamir Secret sharing (continued)

...for some public prime P. Any t sharers can reconstruct $F(x)$ & hence determine S by polynomial interpolation mod P, but $t-1$ sharers cannot. "Linear." Immaculate shared secrets S can be got by having each sharer generate own random secret, then (acting as dealer) deal it out, & then the sum of all of them is S...

As described, scheme vulnerable to cheating dealers (who distribute bogus shares & thus do not really reveal their secret) or cheating sharers (who "reveal" bogus shares to learn the secret while honest players do not). "Verifiable" secret-sharing schemes [Gennaro-Rabin^2, Hirt-PhD] don't have those weaknesses. They require dealer to commit secret before dealing it out, & commit to all the shares he deals out, & ZK-prove the share-commitments correspond to the committed secret; also require the sharers who decide to reveal their shares, to open the share commitments, thus proving share validity.

Threshold-t multiparty cooperative decryption

Decryption exponent K is constant term $P(0)$ of a degree-$(t-1)$ polynomial where decryptor j knows $K_j = P(j)$ but nobody individually knows $P(0)$. Then $P(0)$ deducible by Lagrange polynomial interpolation from $\geq t$ values of $P(x)$. Lagrange interp. is weighted sum $K = \sum_j K_j L_j$ (weights L_j =Lagrange interp. coefficients= public integers); can do exponentiation to power K via

$$x^K = x^{\sum_j L_j K_j} = \prod_j (x^{K_j})^{L_j}.$$

Each decryptor j should broadcast NIZK-proofs he really is exponentiating with his correct private exponent K_j. Note K never learned by anyone.

Plaintext equality test (PET)

Let $(\alpha, \beta) = (g^r, M_1 h^r)$ and $(\gamma, \delta) = (g^s, M_2 h^s)$ be two ElGamal ciphertexts (where r and s are random and g, h public group elements). We wish to test whether $M_1 = M_2$ without revealing r, s, M_1, M_2. Divide: $(\alpha/\gamma, \beta/\delta) = (g^{r-s}, 1h^{r-s})$. Get ElGamal encryption of $1 = M_1/M_2$?

Do cooperative ElGamal decrypt of zth power (z random, immaculate shared secret, $0 < z < P$) of this; note $1^z = 1$ but $M^z \neq 1$. (As usual all parties broadcast ZK proofs they are exponentiating with their correct secret exponents.)

ZK-proofs of ballot validity and interval membership

1. Yes-no election: valid vote is encryption of "1" or "0." Voter could provide an ORed ZK-proof that some ElGamal cryptotext $(g^r, h^r k^M)$ encrypts either $M = 1$ *or* $M = 0$, but not revealing which. (We already showed how to do these component proofs.)

2. If votes consist of integers in a range $[0, 2^b - 1]$, i.e. b-bit integers, the voter could simply provide the elementwise product of b ElGamal 2-tuples,

$$(g^r, h^r k^M) = \prod_{j=0}^{b-1} (g^{r_j},\, h^{r_j} k^{2^j M_j})$$

where $M = \sum_j 2^j M_j$ & each M_j =one-bit, proved as before.

3. If 55 possible legal votes, then need ZK-proof of membership in the integer interval [0, 54].

Correction of common myth about Boudot & interval membership

F.Boudot discussed more general and supposedly more efficient (for large b) interval-membership ZK-proofs than this simple procedure, and also allowing other interval sizes than powers of 2. But his "more efficient" procedure actually is "less efficient" and "more complicated" than this, because Boudot's depends on assumption integer factoring hard, while we depend on assumption discrete logarithms in EC groups hard. So we can use much shorter key lengths to get same security, causing just *one step* in Boudot's method to take more work than our *entire procedure* – Boudot's methods having fewer "steps" is irrelevant. However I [http://math.temple.edu/∼wds/homepage/works.html #80] pointed out & repaired this error by devising ECC replacements for Boudot's ZK-proof components.

Mixnets (=several consecutive Mixers)

"Mixer" inputs N encrypted data and outputs same N items, in shuffled order & re-encrypted. Gives ZK-proof he did that, but not revealing shuffling perm or re-encryption transformations.

Mixnet literature complicated and/or flawed. Now outline simple & good-enough linear-work scheme [see picture]:

1. Shuffler: $C \leftarrow A \rightarrow B$. (Each letter is N data.)
2. Verifier presents random challenge-seed κ.
3. Shuffler uses κ as pseudo-random seed to generate (in standard, crypto-strong way) 2-coloring of C with $\lfloor N/2 \rfloor$ & $\lceil N/2 \rceil$ elements. Publishes the coloring; reveals re-encryptions used to generate the C's from corresponding A's (reveals correspondences) & similarly to generate the B's that come from C's.

Cheating shuffler produced m exceptional (unshuffled or wrongly-encrypted) elements? Detection chance$\geq 1 - 2^{-m}$.

Hashes of secret credentials

Suppose σ is voter's secret credential. Given ElGamal encryption $(g^r, h^r\sigma)$ of σ: want to produce a standard $\text{hash}_z(\sigma)$, ZK-prove we did, but not reveal σ to anybody. Later might want to do again but using different hash-key z so that don't get the same hash of the same σ on this 2nd run.

Let $h = g^\ell$ where ℓ, z, ℓz are immaculate shared secrets. Method:

1. Compute $(g^r)^{\ell z} = h^{rz}$ from the first tuple entry;
2. Compute $(h^r\sigma)^z = h^{rz}\sigma^z$ from the second tuple entry;
3. Divide to get σ^z; (but nobody knows σ or z)
4. Output first 50% of σ^z's bits to get deterministic $\text{hash}_z(\sigma)$.

Cooperating sharers can exponentiate to shared-secret exponents without anybody ever learning the shared secrets or σ.
(Broadcasted ZK-proofs of exponentiation validity of course.)

"Coercion resistance" [JCJ]

"We allow adversary to demand coerced voters vote in specified way, abstain from voting, or disclose secret keys. Scheme coercion-resistant if it's infeasible for adversary to determine whether a coerced voter complies with demands."

Double-edged sword: voter cannot prove/disprove he *voted*. (Immediately knows whether his vote appeared on BB, & can prove did/didn't *register*.) Hence: voter can't be coerced, *but* vulnerable to "EA discards 10% of votes from Black Florida counties." However: if each voter submits *many* votes (which is legal), & to *several* mistrustful EAs, & tries again at new EA if vote not posted, that attack less effective.

Votes are "Additive" or "Anonymizable"?

Anonymizable: your vote unlikely to be unique. "Instant runoff voting": non-additive. Additive⇒Anonymizable: bit-splitting.

Simple Mixer scheme - Problems & Fixes

Problem: As described, was not truly zero knowledge.
Fix: prove each C corresponds to *either* of *two* A's.

Problem: Bogus_m proofs accepted with prob. 2^{-m}.
Fix: Provide K *different* proofs: then 2^{-Km}. Also can do "fractional" proof (faster but less secure): same but $0 < K < 1$. Usually $K = 0.02$ should be good enough in practice; then very fast.