Cards vs. Shuffling

Michael Stay

msta039@ec.auckland.ac.nz

for

Computer Science 725

Clark Thomborson and Jim Goodman

6 June, 2003

In this paper, we review the cryptography necessary to implement two different approaches to electronic voting: the card-based multiparty computation approach of den Boer and others, and Neff's verifiable secret shuffle zero-knowledge proof. We present a variant of the shuffle that uses one-way accumulators. While the card protocols are much more flexible, they are less efficient than Neff's specialized scheme and lack built-in authentication and verifiability; however, we also illustrate how to use a zero knowledge proof in addition to a card protocol to grant the ability to audit the result.

1. Introduction. There are many problems to solve in the area of electronic voting. Some are political; others sociological; here we consider some problems that can be mitigated through the use of cryptography. We examine two different approaches to electronic voting, the card-based approach of den Boer and others, and Neff's verifiable secret shuffle.

2. Proof of membership via the tensor product. Only authorized voters should participate in an election. Benaloh and de Mare [BdM93] suggested one way of proving membership without exposing the identities of other members of the group:

1. A third party generates a modulus $n = pq$ and a number g that generates a large group and reveals (n, g) to the players.
2. The players convert their public keys into bit strings and compile a list.
3. Each player i calculates the product p_i of all the *other* players' public keys, and calculates $y_i = g^{Pi} \bmod n$.
4. The group calculates the product q of all the players' public keys and publishes the value $z = g^q \bmod n$.

Now, when Alice wants to prove membership, she takes her accumulated database y and her public key A and shows that $y^A = z$. This works because it doesn't matter in what order you multiply together the identities; the tensor product is commutative.

An unauthorized Ulrich with public key U would need to find the Uth root of z mod n for his database part. Since taking roots mod n is hard if one doesn’t know its factors, Ulrich can't fake an authorization.

3. Neff's contribution. The problem with using the one-way accumulator to establish a voter's identity is that the signed votes are tied to the voter—votes aren't anonymous. Andrew Neff came up with a way of proving that the key used to sign a vote belongs to a set of authorized voters, but that doesn't reveal which voter.

3.1. Zero Knowledge Proofs. The proof technique that Neff uses is called a *zero-knowledge proof* (ZKP). An ZKP allows a prover Peggy to convince a verifier Victor that she knows a secret without revealing it. These tend to be long and rather complicated; for a few examples beyond the one illustrated below, see [S96].

3.2. ElGamal Cryptosystem. Both of the voting protocols to be illustrated in this paper make use of Taher ElGamal's system [E84] either directly or indirectly. Here's how it works:

1. Alice wants to send Bob a message M. She and Bob agree in public on a large prime p and a generator g for which it is difficult to compute the discrete log mod p to the base g.
2. Bob chooses a random x, and sends
   ($y = g^x$)
   to Alice.
3. To send the secret M, Alice chooses a random k and sends
   ($M \oplus y^k$, g^k), where $\oplus$ is XOR,
   to Bob.
4. Bob calculates ($M \oplus y^k$) $\oplus$ $g^{kx} = M \oplus y^k \oplus y^k = M$.

In essence, the sender picks a new public key and base such that the secret key remains the same. Neff used that property in his shuffle protocol.

3.3. Neff's Simple k-Shuffle.

1. When voting with Neff's shuffle, all the voters go and register; during the registration, they submit an ElGamal public key. Everyone uses the same value of p and g.
2. Once registration has closed, a group of representative voters v_i (perhaps a few each from the press, each party, business, religious leaders, etc.) comes together and each one takes a turn blinding the keys and producing a new base g_i such that

the private keys all remain the same. Specifically, the old keys raised to the c power equal the new keys raised to the d power for some c, d.

If that was all that happened in the protocol, a single player could cheat by replacing the keys with random values or with the keys of his unauthorized friends. So the next steps in the protocol are:

3. The other voter representatives pick a random challenge t and give it to v_i as a challenge.
4. v_i commits to two blinding values, $C = g^{ct}$ and $D = g^{dt}$.
5. All the voter representatives divide by C the set of keys v_i received and divide by D the set of keys v_i produces.
6. All the voter representatives compute the tensor product A of all the keys v_i received and an equal number of C's; also the tensor product B of all the keys v_i published and an equal number of D's.
7. v_i proves that he knows c, d for which $A^c \equiv B^d \pmod{p}$.
8. C becomes the new base for the keys.

If not for the challenge in step 4, v_i could replace all the keys with values having the same tensor product modulo p, which is easy to compute.

3.4. A variant. We note that discrete log mod n is at least as hard as discrete log mod p, since p is just a special case of n. One could use a variant of ElGamal with a composite n and a maximal order g; in this case, the security of the encryption doesn't depend on the factors of n remaining secret. If, however, we assume that they remain secret until after the shuffling is done, we can do the following. Instead of executing steps 3-7 above, each representative

3. computes the one-way accumulation A of the keys he receives,
4. blinds them by raising to a secret power x,
5. computes and published the accumulation B of the new keys,

6. publishes a proof that he knows x for which $A^x \equiv B \pmod{n}$,
7. and announces g^x as the new base.

Each representative after the first in the group can use the same accumulation that his predecessor published, cutting the number of exponentiations almost in half and obviating the need for the group to perform the calculations involved in the challenges. The downside to this approach is the increased length of the public keys—they're twice as long for the same security, so the workload is almost the same.

The reason that the factors of n have to stay secret until after the shuffling is that the security of the accumulator depends on the inability to take roots over the modulus.

4. Secure multiparty computation via oblivious transfer. *Secure multiparty computation* is a cryptographic technique by which multiple players can compute a function of their inputs without revealing anything about the inputs other than what can be deduced from the output. The most straightforward way to perform the computation is via a cryptographic protocol called *oblivious transfer*.

4.1. Oblivious Transfer. Oblivious transfer was invented by Michael Rabin [R81]; it allows Alice to let Bob to pick one of two secrets without knowing which one he picked. One implementation uses ElGamal:

1. Alice and Bob agree on a large prime p and a generator g for which it is difficult to compute the discrete log mod p to the base g.
2. Bob chooses a random x, a random bit r, and sends
   $(y_0 = 2^r g^x, y_1 = 2^{r-1} g^x)$
   to Alice. Note that Bob only knows the discrete log of y_r, namely x.
3. To send the secret, Alice chooses a random k and sends
   $(M_0 \oplus y_0^k, M_1 \oplus y_1^k, g^k)$, where $\oplus$ is XOR,
   to Bob.
4. Bob calculates $(M_r \oplus y_r^k) \oplus g^{kx} = M_r \oplus y_r^k \oplus y_r^k = M_r$.

4.2. Multiparty computation. To compute a function f:{0,1}²→ {0,1}of Alice's secret bit A and Bob's secret bit B, they follow the protocol above, but Bob chooses $r=B$ and Alice chooses $M_i = f(A, i)$. Then Bob gets $M_B = f(A, B)$, which he announces to Alice. Neither one can find out what the other party's secret is except for what they can deduce from the function and their own input. This protocol works when the players are "semi-honest": Alice computes the right function, and Bob reveals the right result to Alice, but both keep track of everything they see.

To compute larger functions, Bob simply refrains from announcing the result until the end of the computation, carrying out the protocol for every gate involving both his bits and Alice's. It's clear that for large circuits, there will be many, many exponentiations. Exponentiation is a relatively slow process, and researchers have come up with various ways of making the solution more efficient.

5. Card protocols. Bert den Boer's now classic paper [B89] introduced the secure computation of an AND gate using two kinds of cards. It requires the same number of exponentiations as the previous protocol, but as we'll see later, it can be improved.

Alice and Bob take three identical decks of cards and remove three copies of the 2 of spades and two copies of the 2 of hearts. The rest of the deck is set aside. One of the spades is laid down on the table. Alice and Bob each get one heart and one spade. If Alice wishes to encode a '1', she places the spade beneath her heart; otherwise, the spade is on top. She places the cards face down on top of the spade on the table. Bob does the mirror image: if he wants to encode a '1', the spade goes on top. He places his cards face down underneath the deck on the table. Next, each privately cuts the deck. Finally, the cards are turned face-up. If the cards are a cyclic shift of (♥, ♠, ♠, ♠, ♥), then the result is a '1'; otherwise, it's a '0'.

In this protocol, the table is a third player: it provides some way to hide the face of the card. den Boer showed how one can do the same thing mathematically with Jacobi symbols.

5.1. Legendre symbol. The Jacobi symbol is a composite version of the Legendre symbol. The Legendre symbol tells whether a number has a square root modulo a prime p or not. It's based on Fermat's little theorem:

$$b^{p-1} \bmod p \equiv 1,\ b \neq 0 \bmod p.$$

Since all $p > 2$ are odd, p-1 is even, and

$$b^{(p-1)/2} \bmod p \equiv \pm 1,\ b \neq 0 \bmod p.$$

$b^{(p-1)/2} \bmod p$ is the Legendre symbol. If it is positive, b can be written as

$$b \equiv a^2 \bmod p;$$

that is, b is a *quadratic residue modulo* p. If it is negative, there is no number a such that $a^2 \bmod p \equiv b$; that is, b is a *nonresidue modulo p*.

5.2. Jacobi symbol. The Jacobi symbol for a number $n = pq$ is simply the product of the Legendre symbols mod p and mod q. It can also be computed quickly in a way similar to the greatest common divisor function if one does not know p and q. If the Jacobi symbol of b is –1 modulo n, then b is a nonresidue mod n. However, if the Jacobi symbol is 1, that doesn't necessarily mean it's a residue! This is because a number can be a nonresidue modulo both p and q at the same time. In this case, the product of the Legendre symbols is

$$(-1)(-1) = 1.$$

5.3. den Boer's protocol. den Boer's protocol uses Jacobi symbols like this:

1. Charles generates a modulus $n = pq$ and a number K such that K is a nonresidue but the Jacobi symbol mod n is +1. He then sends
   (n, K)
   to Alice and Bob.
2. Alice picks a secret random number r_A and squares it. Bob picks a secret random number r_B and squares it. Each then encodes the cards as follows:
   ♥ = r^2, ♠ = K r^2.
   Note that the other player can't tell whether it's a heart or a spade—both have Jacobi symbol +1.
3. They place the cards as before, without showing Charles.
4. They cut the deck without showing Charles; this time it doesn't matter if Alice and Bob both see.
5. They send the deck to Charles.
6. Charles checks the Legendre symbol of the result cards modulo p and decodes the symbols as follows:
   +1 = ♥, -1 = ♠.
   No one else can do this because it's hard to factor n.
7. Charles compares the decoded cards to (♥, ♠, ♠, ♠, ♥) and announces the result to Alice and Bob.

5.4. Some improvements. den Boer's protocol has a major shortcoming: Alice and Bob have to ask Charlie every time they need to compute the AND of their bits. Crepeau and Killian [CK93] designed a probabilistic protocol requiring 10 cards and 2.5 tries on average that presents the output in a committed form and allows one to duplicate bits without revealing them. This way, Alice and Bob can compute an entire circuit and only need to ask Charlie to decrypt the output. Anton Stiglic [S01] improved their protocol to use 8 cards per gate and an average of 2 tries. If we choose the circuit to be a simple majority circuit, we can use this protocol for voting: each player encodes his vote as a bit

string; the circuit picks the winner, and Charlie reveals the winner's name to the players—all with only a single exponentiation for each output bit.

6. Auditing. Finding a suitably honest Charlie for a major election is a daunting task—how can one know that Charlie didn't just pick the result himself? If he reveals *p* so that others can check the Legendre symbols, the players will be able to tell what cards were used because the r chosen is unique to the player. The vote would no longer be anonymous. We would like the ability to audit Charlie's calculations, and can do so if Charlie provides a ZKP that the Legendre symbols are as he published them.

[L03] presents a method of constructing short (subquadratic) zero-knowledge proofs for lists of integers satisfying certain Diophantine relations. Since the computation of the Legendre symbol falls into this class of relations, this allows Charlie to prove he computed the Legendre symbols correctly.

6.1. Authentication. Neff's protocol authenticates the voters as belonging to a known, registered group. In the card protocols, one can implement a circuit to verify a signature on the input, but it becomes inefficient when compared to specialized protocols like Neff's, since there are multiplications and cuts for each Boolean gate in the circuit.

7. Conclusion. We've seen that specialized voting protocols like Neff's shuffle are more efficient, but less flexible than generic multiparty computations like the card protocols. We can add auditing to a card protocol to make it verifiable and comparably as efficient as Neff's protocol at the expense of trading user authentication for keeping the votes secret. We presented our own variant of Neff's shuffle that uses composite ElGamal. Finally, we can add authentication to the card protocol, but only at a great cost in efficiency.

[B89] den Boer, Bert, "More efficient match-making and satisfiability." *Advances in Cryptology – EuroCrypt '89*, LNCS 434, J. Quisquater and J. Vandewalle, Eds. Springer-Verlag, Berlin, 1989. pp. 208-218.

[BdM93] Benaloh, J.C. and M. de Mare, "One-Way Accumulators: A Decentralized Alternative to Digital Signatures." *Advances in Cryptology – EuroCrypt '93*, LNCS 765, T. Helleseth, Ed. Springer-Verlag, Berlin, 1993. pp. 274-285.

[CK93] Crepeau, C. and J. Kilian, "Discreet solitary games." *Advances in Cryptology – Crypto '93*, LNCS 773, D. Stinson, Ed. Springer-Verlag, Berlin, 1993. pp. 319-330.

[E84] ElGamal, T., "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." *Advances in Cryptology—Crypto '84*. Springer-Verlag, Berlin. 1985. pp. 10-18.

[L03] Lipmaa, Helger, "On Diophantine Complexity and Statistical Zero-Knowledge Arguments." Electronic pre-print, IACR E-print server, 2003. http://eprint.iacr.org/2003/105

[R81] Rabin, M. O., "How to Exchange Secrets by Oblivious Transfer." *Technical Memo TR-81*. Aiken Computer Laboratory, Harvard University, 1981.

[S96] Schneier, Bruce, *Applied Cryptography, 2nd Edition*. John Wiley and Sons, Inc. New York. 1996. p. 548-549.

[S01] Stiglic, Anton, "Computations with a deck of cards." *Theoretical Computer Science*, vol 259(1-2). 2001. pp. 671-678.