コンピュータセキュリティ 14－7
(2001. 7. 25)

秘密カウンタ

菊池 浩明　中里 純二　中西 祥八郎
東海大学電子情報学部
259-1292 神奈川県平塚市北金目 1117
TEL: 0463-58-1211 内線 4164, FAX: 0463-59-4014
Email: {kikn,nakazato}@ep.u-tokai.ac.jp

あらまし 本論文は，1か0かの秘密を漏らさないまま，公開検証可能な方法で足し算を実行するプロトコルを提案している．提案方式は，従来の膨大な帯域を消費し何回にも及ぶラウンドを必要としたマルチパーティプロトコルとは異なり，非対話的な定数回のラウンドと単純で送信者にも集計者にも検証可能な計算処理を必要とする．提案方式は，[2] に基づいている．提案方式は，投票者が投票用紙を分散された公開鍵について暗号化して投票し，集計者が賛成投票数をわからないまま数えるという秘密投票に応用可能である．

キーワード ゼロ知識証明，電子投票，秘密関数計算

Oblivious Counter

Hiroaki Kikuchi　Junji Nakazato　Shohachiro Nakanishi
School of Information Technology and Electronics
Tokai University
1117 Kitakaname, Hiratsuka, Kanagawa 259-1292, Japan
Tel: +81-463-58-1211 Ext. 4164
Fax: +81-463-50-2412
Email: kikn@ep.u-tokai.ac.jp

Abstract The paper presents a new protocol for counting 1-bit secrets without revealing if the bit is 1 or 0 in publicly verifiable way. Other than the conventional multi-party protocols that involve enormous number of rounds and huge bandwidth consumption, the proposed protocol, based on the Mix and Match approach [2] in which computations are dealt with ciphertexts, requires a non-interactive constant number of round and simple but verifiable computation for both of sender and counter. The expected application of proposed protocol is an (one-bit) secret voting in which voters cast a ballot encrypted for distributed public key and an oblivious party makes a tally of how many votes are polled.

Key words zero-knowledge proof, electrionic voting, secret function computation

1 Introduction

Oblivious Counter is a counter which takes n ciphertexts of one-bit secret as an input and outputs the $\log n$ ciphertexts which encodes the sum of all secrets. The counter is oblivious party who learn no information about the secrecy from his inputs. The counter has no private information and thus the processes of adding are all publicly verifiable. The corresponding private key is shared among some trusted parties called administrators, who involve at the end of the protocol in order to jointly decrypt the tally.

The indented use of oblivious counter is electronic voting, in which we have two conflicting goals; the privacy of votes and the accountability of parties. (For requirements and styles of voting, see [1]. So far, major efforts have been made for universally and publicly verifiable Mix-net.) The oblivious counter can match the requirements, i.e., each voter can casts a ciphertext of his/her private vote to the oblivious counter without care of the private vote being compromised or discarded. The double-counting and altering are prevented by an additional verification protocol.

Our idea is based on the protocol "Mix and Match" presented by Jakobsson and Juels[2]. The Mix-and-match is a multiparty protocol in which an arbitrary function is computed without revealing private input values. Other than the GMW multiparty schemes, the Mix-and-match manipulates ciphertexts directly without being secret shared. New building block is introduced; Plaintext Equally Test ($\mathcal{PET}$) which examine two ciphertext to be identical or not without revealing plaintexts.

The naive approach of Mix-and-Match to electronic voting is that El Gamal ciphertexts of votes (say $b_i = a$ for yes; $b_i = a^{-1}$ for no) are simply multiplied and then published. The resulting ciphertext contains the total number of active votes S ($\sum E[b_i] = E[\prod b_i] = E[a^S]$). The approach satisfies the privacy of votes and publicly verifiability. However, to open the tally, we have to examine all possible S even after it is decrypted. Letting n be the possible number of votes, the naive protocol takes a cost of $O(n)$, which is far practical.

Instead, we present $O(\log n)$ solution in this paper. Our idea is based on n-bit (up) counter which takes one bit input and outputs the n-bit number of inputs. The PET can be used to design the arbitrary counter, however, it requires enormous number of interaction between counter and administrators who have the shared private key. Instead, we use internal states as carry bits in conjunction with proof of disjunctive knowledge presented in [4]. Our contribution is to present up-counter protocol without AND.

2 Preliminaries and Building Blocks

2.1 Models and Assumptions

We have three parties; *voters* who has an one bit secret, say 1 or 0, and let denote by N the number of total voters, *counter*, an oblivious party who keeps a tarry of votes without revealing the result of count until the time to open the ballot boxes comes, and *administrators* who are trusted parties sharing a private key and jointly decrypt the result of count when the time to open voting box comes. We assume that some of administrators may be faulty but not more than $2/n$ can corrupt, namely, any appropriate threshold schemes such as [3] can be used to recover the corresponding plaintext in a secure way. All players agree the security parameter and the public key securely and jointly generated by the administrators. We assume an authenticated broadcast channel by which the counter can effectively publish the proof of every vote being counted in the tally, and the common public-key infrastructure for listing all legitimate voters.

2.2 Building Blocks

2.2.1 El Gamal Cryptosystem

Let $p = 2q+1$ and $\mathcal{G}$ be the set of multiplicative group of order q in Z_p^*. Let g be an element of $\mathcal{G}$. All arithmetic operations are done in modulo p, unless otherwise stated. An El Gamal encryption of message m with public key $y = g^x$ is of form $E_a[m] = (M, G) = (my^a, g^a)$, where a is a random number chosen from Z_q. To decrypt the ciphertext (M, G), we use the corresponding private key x to compute $M/G^x = mg^{xa-ax} = m$. The El Gamal cryptosystem is indistinguishable under the Decision Diffie-Hellman (DDH) assumption over $\mathcal{G}$. Namely, given two ciphertexts $E[m]$ and $E[m']$, no one can learn if $m = m'$ or not. This property allows us to construct an oblivious computation in later section.

The El Gamal cryptosystem satisfies a *homomorphism* under a multiplication of two ciphertexts, i.e., given two ciphertexts $E_a[m_a] = (M_a, G_a)$ and $E_b[m_b] = (M_b, G_b)$, the products of the ciphertexts for each element yields a new valid ciphertext $(M_aM_b, G_aG_b) = (m_am_by^{a+b}, g^{a+b}) = E_{a+b}[m_am_b]$. We write this as $E[M_a] \times E[M_b]$. Obviously, the homomorphism holds not only under a multiplication but also under a division and under a raising to exponent, that are used to blind a result of Plaintext Equality Test in [2].

One more good property of the El Gamal cryptosystem is that it can be extended to use the threshold schemes [3]. A private key is jointly generated by a collaboration of t honest parties (key holders) out of n and distributed among them using $(t-1)$-degree random polynomials $f(x)$ as $f(1), f(2), \ldots, f(n)$. To decrypt a ciphertext provided with the public key $y = g^{f(0)}$, i-th party publishes $G^{f(i)}$ for $i = 1, \ldots, t$, and then computes $G^{f(1)\gamma_1} \ldots G^{f(t)\gamma_t} = G^{f(0)}$ where γ_i is the LaGrange coefficient for i. For the sake of verifiability, we can have the key holders proving that they follows the protocol correctly in the zero-knowledge proof of $f(i)$ such that $G^{f(i)}$. See [3] for details.

2.2.2 Various Protocols for Proof of Knowledge

We will use a proof of knowledge of private input to the counter, which is based on the variation of disjunctive and conjunctive proofs of knowledge in [4].

Conjunctive Proof of Knowledge: Let $g_1, g_2 \in \mathcal{G}$. By $PK\{(\alpha) : y_1 = g_1^\alpha \wedge y_2 = g_2^\alpha\}$, we denote a proof of knowledge of discrete logarithms of elements y_1 and y_2 to the bases g_1 and g_2. Picking random numbers r_1 and $r_2 \in Z_q$, a prover sends $t_1 = g_1^{r_1}$ and $t_2 = g_2^{r_2}$ to a verifier, who then sends back a random challenge $c \in \{0,1\}^k$. The prover shows $s = r - cx \pmod q$, which should hold both $g_1^s y_1^c = t_1$ and $g_2^s y_2^c = t_2$.

Disjunctive Proof of Knowledge: We denote by $PK\{(\alpha, \beta) : y_1 = g^\alpha \vee y_2 = g^\beta\}$ to mean a proof of knowledge of one out of the two discrete logarithms of y_1 and y_2 to the base g. Namely, the prover can prove that he knows a secret value under which either $y = y_1$ or $y = y_2$ must hold without revealing which identity was used. Without loss of generality, we assume that the prover knows α for which $y = g^\alpha$ holds. The prover uniformly picks $r_1, s_2 \in Z_q$ and $c_2 \in \{0,1\}^k$ and sends $t_1 = g^{r_1}$ and $t_2 = g^{s_2} y_2^{c_2}$ to the verifier, who then gives a random challenge $c \in \{0,1\}^k$, where k is a security parameter. On receiving the challenge, the prover sends $s_1 = r_1 - c_1\alpha \pmod q$, s_2, c_1 and c_2, where $c = c_1 \oplus c_2$. The verifier can see if the prover is likely to have the knowledge by testing both $t_1 = g^{s_1} y_1^{c_1}$ and $t_2 = g^{s_2} y_2^{c_2}$ with provability $1 - 2^{-k}$. Note that the same test can be used when t_1 and t_2 are prepared for the other knowledge β.

3 Oblivious Counter

3.1 1-bit (Half) Adder

Let $V = \{1, -1\}$ be a set of truth values meaning boolean values *false* (0) and *true* (1). Let a and b be boolean variables in V. We denote by capital letters the ciphertexts of boolean variables, e.g., $A = E_r[a]$ and $B = E_s[b]$. A *half adder* is a state-full circuit that takes two inputs, a and b, and produces *sum* s and *carry* c as outputs computed by

$$s = a \oplus b, \quad c = a \wedge b,$$

and the internal state a will be updated when (edge-triggered) clock pulse arrives, we write this by $a(t+1) = s(t)$ where t indicates the synchronous (discrete) time.

In consequence of the multiplicative homomorphism of El Gamal cryptosystem, the function EXORing of two plaintexts, a and b, is easily implemented by simply multiplying the ciphertexts A and B. Table 1 demonstrates the result of multiplication, which shows the truth table of EXOR.

Table 1: Truth Table of EXOR

$A \times B = S$	$a \oplus b = s$
$E[1] \times E[1] = E[1]$	$0 \oplus 0 = 0$
$E[1] \times E[-1] = E[-1]$	$0 \oplus 1 = 1$
$E[-1] \times E[1] = E[-1]$	$1 \oplus 0 = 1$
$E[-1] \times E[-1] = E[1]$	$1 \oplus 1 = 0$

The carry c (AND) is, however, hard to realize since the set of logical operations EXOR and negation, which is done by EXORing 1, is not sufficient for achieving the functionally completeness. The Mix and Match approach [2] is a solution to this problem, in which a truth table of AND is blinded and row-wise mixed and then the plaintext equality test ($\mathcal{PET}$) is used to figure out the *match* of the corresponding row for given input ciphertexts with the help of distributed private key dealers. Rather than involving trusted parties and many other players (MIXs), we would like to make our protocol non-interactive, i.e., avoiding the involvement of other parties except the voter who has the private input b and the counter who has the private state a.

3.2 2-bit Counter

In this section, we show that the oblivious 2-bit counter is feasible without using the logical AND.

A voter has an one-bit secret b and wishes to have the counter add b to the oblivious tally without revealing if b is 1 or 0. First, the voter casts a pair of ciphertexts (B, C) computed by

$$(B, C) = \begin{cases} (E[b], E[1]) & \text{if } b = 1, \\ (E[b], E[A]) & \text{if } b = -1, \end{cases}$$

where A is the ciphertext of a and $E[A]$ is the corresponding re-encryption defined as $E[A] = A \times E[1]$. The element B contains one-bit secret b to be added and the element C gives the output carry digit. Note that no one can distinguish $E[A]$ from $E[1]$ from the assumption of semantically secure El Gamal cryptosystem.

Second, in order to prevent faulty voters from casting invalid pairs of ciphertexts, in particular, $(E[-1], E[1])$ and $(E[1], E[-1])$, we require voters to provide along with the B, C the proof of disjunctive knowledge

$$PK\{(u, v) : B = E_u[1] \wedge C = E_v[1]\} \vee \{(u', v') : B = E_{u'}[-1] \wedge C = E_{v'}[A]\}.$$

We present the proof of knowledge based on zero-knowledge protocols in [4].

Given $B = (M_B, G_B)$ and $C = (M_C, G_C)$, let us define

$$\begin{array}{ll} M_{B1} = M_B, & G_{B1} = G_B, \\ M_{C1} = M_C, & G_{C1} = G_C, \\ M_{B2} = -M_B, & G_{B2} = G_B, \\ M_{C2} = M_C / M_A, & G_{C2} = G_C / G_A, \end{array}$$

where (M_A, G_A) is the current ciphertext A. Protocol specified in Table 2 ensures that the pair of ciphertexts (B, C) is either $E[1] = (M_{B1}, G_{B1})$ and $E[1] = (M_{C1}, G_{C1})$ or $E[1] = (M_{B2}, G_{B2})$ and $E[1] = (M_{C2}, G_{C2})$. The technical details of the protocol are omitted.

On receiving (B, C) and additional proof of knowledge from a voter, the counter computes the sum of input b and the previous result encoded by (A_2, A_1) (A_2 is MSB), as for each digits

$$S_1 = A_1 \times B, \quad S_2 = A_2 \times C.$$

The internal states A_1 and A_2 subsequently are updated by

$$A_1(t+1) = S_1(t), \quad A_2(t+1) = S_2(t).$$

An initial state for A_i is $E[1]$ additionally proved in some manner. Since any computation involved in the counter does not reveal private information of input nor the tally, the verification of procedures can be publicly done. Any party who is interested in an accountability of the counter can easily check the consistency by comparing the previously published sum $A_2(t-1), A_1(t-1)$ and the updated sum $A_2(t), A_1(t)$.

Table 3 demonstrates the states transition when "active" voter who has $b = -1$ comes every even clock. The carry digit c_1 is implemented by a duplication of the third row headed by a_1 but the duplication happens only when $b = -1$. For example, the $c_1 = -1$ at clock $t = 4$ comes from the a_1 at the same clock but the duplication at $t = 2$ does not alter the internal state s_1 at all. Hence, it can properly deal with cases when multiple "unactive" votes cast in succession.

3.3 n-bit Counter

In switching circuit theory, the 2-bit counter is known as a half-adder for which n-bit counter can be formed by cascading n adders. The carry bits ripple from least significant bit to most significant bit. Our construction of 2-bit counter, however, requires a carry bit being duplicated from the previous internal state (A_1). Thus, the n-bit extension is not trivial.

The idea is having voter cast with all subsequent carry bits. For simplicity, let us consider a case $n = 3$ where we wish to have C_2. The second carry bit C_2 is provided by voter as

$$C_2 = E[c_1 \wedge a_2] = \begin{cases} E[1] & \text{if } b = 1, \\ E[A_2'] & \text{if } b = -1, \end{cases}$$

where A_2' is defined as follows:

$$A_2'(t+1) \quad = \quad S_2'(t)$$

Table 2: Protocol for Proof of Knowledge $PK\{(u,v) : B = E_u[1] \wedge C = E_v[1]\} \vee \{(u',v') : B = E_{u'}[-1] \wedge C = E_{v'}[A]\}$

Step	$(b=1)$	$(b=-1)$
1	The prover (voter) picks random elements $r_{B1}, r_{C1}, z_{B2}, z_{C2} \in Z_q$ and $d_2 \in \{0,1\}^k$ if $b = 1$; otherwise $r_{B2}, r_{C2}, z_{B1}, z_{C1} \in Z_q$ and $d_1 \in \{0,1\}^k$, on which he computes the followings and sends to the verifier.	
	$t_{B1} = g^{r_{B1}}$	$t_{B1} = g^{z_{B1}} G_{B1}^{d_1}$
	$w_{B1} = y^{r_{B1}}$	$w_{B1} = y^{z_{B1}} M_{B1}^{d_1}$
	$t_{B2} = g^{z_{B2}} G_{B2}^{d_2}$	$t_{B2} = g^{r_{B2}}$
	$w_{B2} = y^{z_{B2}} M_{B2}^{d_2}$	$w_{B2} = y^{r_{B2}}$
	$t_{C1} = g^{r_{C1}}$	$t_{C1} = g^{z_{C1}} G_{C1}^{d_1}$
	$w_{C1} = y^{r_{C1}}$	$w_{C1} = g^{z_{C1}} M_{C1}^{d_1}$
	$t_{C2} = g^{z_{C2}} G_{C2}^{d_2}$	$t_{C2} = g^{r_{C2}}$
	$w_{C2} = y^{z_{C2}} M_{C2}^{d_2}$	$w_{C2} = y^{r_{C2}}$
2	The verifier randomly picks a challenge d from $\{0,1\}^k$ and sends to the prover.	
3	The prover computes $d_1 = d \oplus d_2$ and sends d_1, d_2 and	The prover computes $d_2 = d \oplus d_1$ and sends d_1, d_2 and
	$z_{B1} = r_{B1} - d_1 u$	z_{B1}
	z_{B2}	$z_{B2} = r_{B2} - d_2 u$
	$z_{C1} = r_{C1} - d_1 v$	z_{C1}
	z_{C2}	$z_{C2} = r_{C2} - d_2 v$
4	The verifier accepts if $d = d_1 \oplus d_2$ and	
	$g^{z_{B1}} G_{B1}^{d_1} = g^{r_{B1} - d_1 u + u d_1} = t_{B1}$	$g^{z_{B1}} G_{B1}^{d_1} = t_{B1}$
	$y^{z_{B1}} M_{B1}^{d_1} = y^{r_{B1} - d_1 u + u d_1} = w_{B1}$	$y^{z_{B1}} M_{B1}^{d_1} = w_{B1}$
	$g^{z_{B2}} G_{B2}^{d_2} = t_{B2}$	$g^{z_{B2}} G_{B2}^{d_2} = g^{r_{B2} - d_2 u + u d_2} = t_{B2}$
	$y^{z_{B2}} M_{B2}^{d_2} = w_{B2}$	$y^{z_{B2}} M_{B2}^{d_2} = y^{r_{B2} - d_2 u + u d_2} = w_{B2}$
	$g^{z_{C1}} G_{C1}^{d_1} = g^{r_{C1} - d_1 v + v d_1} = t_{C1}$	$g^{z_{C1}} G_{C1}^{d_1} = t_{C1}$
	$y^{z_{C1}} M_{C1}^{d_1} = y^{r_{C1} - d_1 v + v d_1} = w_{C1}$	$y^{z_{C1}} M_{C1}^{d_1} = w_{C1}$
	$g^{z_{C2}} G_{C2}^{d_2} = t_{C2}$	$g^{z_{C2}} G_{C2}^{d_2} = g^{r_{C2} - d_2 v + v d_2} = t_{C2}$
	$y^{z_{C2}} M_{C2}^{d_2} = w_{C2}$	$y^{z_{C2}} M_{C2}^{d_2} = y^{r_{C2} - d_2 v + v d_2} = w_{C2}$

Table 3: State Transition Table of the 2-bit Counter

t	time	1	2	3	4	5	6	7	8	9
b	input	1	-1	1	-1	1	-1	1	-1	1
a_1	$= s_1(t-1)$	1	1	-1	-1	1	1	-1	-1	1
s_1	$= a_1(t) \oplus b(t)$	1	-1	-1	1	1	-1	-1	1	1
c_1	$= a_1(t) \wedge b(t)$	1	1	1	-1	1	1	1	-1	1
a_2	$= s_2(t-1)$	1	1	1	1	-1	-1	-1	-1	1
s_2	$= a_2(t) \oplus c_1(t)$	1	1	1	-1	-1	-1	-1	1	1
(a_2, a_1)	decimal	0	0	1	1	2	2	3	3	0

$$
\begin{aligned}
S_2'(t) &= C_2'(t) \times A_2'(t) = E[c_2' \oplus a_2'] \\
C_2'(t) &= \begin{cases} E[1] & \text{if } b = 1, \\ E[A_2] & \text{if } b = -1. \end{cases} \\
A_2(t+1) &= S_2(t) \\
S_2(t) &= C_1(t) \times A_2(t)
\end{aligned}
$$

The ballot voter has to send consists of B, C_1, C_2' and C_2. The carry bit C_1 is identical to C in the 2-bit counter protocol. In the protocol, note that the intermediate internal state A_2' and S_2' are used as T-FF (Toggle Flip Flop) which decrease a cycle by one-half for each. We illustrate how 3-bit counter works in Table 4. For simplification, we assume every vote is "active". At the bottom row, we can observe that the second carry can be obtained properly.

The n-bit counter can be naturally extended; the voter fetches the current internal states $A_1, \ldots, A_n$ which have been published and updated for every new vote is added to the counter. Then, the voter computes the ciphertexts of ballot according to his private vote (b) and additional ciphertexts for bits of T-FFs, and sends it with optional proof of disjunctive knowledge of conjunction of ciphertexts for all bits.

In the previous section, we see that the T-FF decreases the cycle by half for each. Hence, with appropriate number of T-FFs, any carry bit c_k can be computed from the one-bit-left carry c_{k-1}. For instance, to implement c_3, we need three additional T-FFs, i.e., a_3', a_3'' and a_3''', and the total of $4 + 4 = 8$ ciphertexts are necessary to cast one bit. In general, n-bit counter requires voter to send the total of 2^{n-1} ciphertexts.

Remark 3.1 *The communication cost for voter is* $O(2^n)$.

The overhead at the counter is considerably small because it has no private information such as private key and thereby it can reduce the cost for operation. But, the update process may alter at most 2^{n-1} internal states.

Remark 3.2 *The computation cost for counter is* $O(2^n)$.

The cost of administrators is the lightest in our protocol. The administrators involved at the begging of the protocol (for key generation) and the end for distributed description of the tally $(A_n, A_{n-1}, \ldots, A_1)$, which gives the total number of "active votes".

Remark 3.3 *The computation cost for administrator is* $O(n)$.

Table 4: State Transition Table of 3-bit Counter

t	1	2	3	4	5	6	7	8
b	-1	-1	-1	-1	-1	-1	-1	-1
a_1	1	-1	1	-1	1	-1	1	-1
s_1	-1	1	-1	1	-1	1	-1	1
c_1	1	-1	1	-1	1	-1	1	-1
a_2	1	1	-1	-1	1	1	-1	-1
s_2	1	-1	-1	1	1	-1	-1	1
c_2'	1	1	-1	-1	1	1	-1	-1
a_2'	1	1	1	-1	1	1	1	-1
s_2'	1	1	-1	1	1	1	-1	1
c_2	1	1	1	-1	1	1	1	-1

4 Conclusion

We presented an oblivious counter protocol which allows us to count up "active votes" without revealing privacy of voters.

References

[1] M. Abe, "Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers," IEICE Trans. Fundamentals, Vol. E83-A, No.7, July 2000.

[2] M. Jakobsson and A. Juels, "Mix and Match: Secure Function Evaluation via Ciphertexts," Proc. of ASIACRYPTO 2000, LNCS 1967, pp. 162-177, 2000.

[3] T. P. Pedersen, "A threshold cryptosystem without a trusted party," *EUROCRYPTO '91*, pp.522-526, 1991.

[4] R. Cramer, I. Damgard, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," *CRYPTO '94*, pp.174-187, 1994.

[5] J. Camenisch and M. Michels, "Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes," *EUROCRYPT'99*, pp. 107-122, 1999.