# Publicly Verifiable Batch Encryption

*Po-Wen Ko*, *Jong-E Hsien*, and *Chien-Yuan Chen*
Department of Information Enginerring,
I-Shou University, Kaohsiung County, Taiwan, 840 R.O.C.
E-mail: {m883317m, m883311m, cychen}@ isu.edu.tw

## Abstract

In this paper, we propose a publicly verifiable batch encryption scheme, which allows a third party to verify some encrypted messages without revealing any information at the same time. These encrypted messages can also be verified by running a single PVE scheme many times. However, it is less efficient than our scheme. According to our analysis, our scheme can reduce the overhead of computations by 66%. Furthermore, our scheme can be applied to batch escrowed e-cash systems.

**Keyword**: public verifiable encryption, escrowed e-cash.

### 1. Introduction

Since its invention [4], verifiable secret sharing (VSS) is a cryptographic primitive to achieve security against cheating participants. In [10], Stadler presented a publicly verifiable encryption (PVE) technique of discrete logarithms, that allows the prover to convince the verifier that the ciphertext C hinds a message m under someone's public key without revealing the message. For example, Alice places a piece of paper in a translucent envelope and sends it to Bob. Bob can verify that there is a piece of paper in the envelope without opening it.

PVE can be used for e-commerce-related cryptographic protocols such as escrow-cryptosystems [7, 8] and fair exchange of digital signatures [1, 2]. In some cases, the sender must send several ciphertexts using the PVE scheme at the same time. Thus, the overhead of computations for running a single PVE scheme to verify these ciphertexts is more expensive. That motivates us to present a publicly verifiable batch encryption to reduce the overhead.

In this paper, we present a new publicly verifiable encryption that allows a trusted entity to verify batch encryption at the same time. For example, if the sender Alice wants to send several ciphertexts $C_1$, $C_2$, …, $C_n$ to a group of participants via the verifier Bob. Bob can verify these encrypted messages without seeing the messages. Instead of running a single PVE scheme for every ciphertext, our publicly verifiable batch encryption (PVBE) scheme is more efficient.

Based on our PVBE scheme, we combine the batch signature [3, 9] to construct batch escrowed electronic cash systems. Batch signatures allow many coins to be withdrawn once. The cost is reduced for verifying many coins together. As introduced in [9], it allows multiple coins can be withdrawn and spent by using batch protocols. In this paper, we will construct a batch escrowed electronic cash system, which allows the user Tom to withdraw coins one time, later he can dispense each coin to other participants via Bob, who acts on behalf of recipients to verify that Tom has indeed sent the e-cash to every recipient.

The remainder of this paper is organized as follows. In the next section we review Stadler's PVE scheme. We present a PVBE scheme in Section 3. Furthermore, our presented PVBE scheme can be applied to e-cash systems in Section 4. Finally, Section 5 concludes the work of this paper.

### 2. Review of Stadler's Scheme

Before the introduction of Stadler's scheme, some parameters should be defined, that will be used throughout this paper.

Let p, q, and r be large primes, where $q = 2 \times r + 1$, $p = k \times q + 1$. Here k is any even number. Let further G be a group of order q generated by g and H be a group of order r generated by h. Then computing discrete logarithms to the bases g or h is difficult.

According to the double discrete logarithm from $x \in Z_r$ to $y \in G$ with bases g and h, we have

$$Y = g^{h^X \bmod q} \bmod p.$$

Here $Z_r$ denotes the ring of integers modulo r.

Stadler' s scheme uses ElGamal' s public key system [5] to encrypt the message and a zero-knowledge protocol to convince the verifier that the ciphertext hinds the message m without revealing the message.

There are three participants: Alice (the sender), Bob (the verifier), and Nancy (the recipient). The private/public key pairs of Alice, Bob and Nancy are ($X_A$, $Y_A = g^{X_A} \bmod p$), ($X_B$, $Y_B = h^{X_B} \bmod q$) and ($X_N$, $Y_N = h^{X_N} \bmod q$), respectively. To encrypt a message $m \in Z_q^*$, where $Z_q^*$ denotes the ring of integers modulo q except zero, Alice randomly chooses $k \in Z_q$ to calculate $A = h^k \bmod q$ and $B = m^{-1} Y_N^{\,k} \bmod q$. Nancy can decrypt the ciphertext (A, B) by computing $m = A^{X_N} / B \bmod q$.

In Stadler' s scheme, Alice sends A, B, Y' to Bob and convinces Bob that the pair (A, B) hinds the message m without revealing the message under a public value $Y' = g^m \bmod p$ by running Protocol ZP1 j times for some integer j. Protocol ZP1 is listed as follows:

**Protocol ZP1:**

**Step 1:** Alice randomly chooses $w \in Z_r$ to calculate

$$t_1 = h^w \bmod q \quad (1)$$

$$\text{and } t_2 = g^{(Y_N)^w} \bmod p. \quad (2)$$

Then she sends Bob t1, t2.

**Step 2:** Bob randomly chooses C, where C =1 or 0, and sends it to Alice.

**Step 3:** After receiving C, Alice computes $R = w - Ck \bmod r$, and she sends R to Bob.

**Step 4:** Finally, Bob checks whether

$$t_1 = h^R A^C \bmod q \quad (3)$$

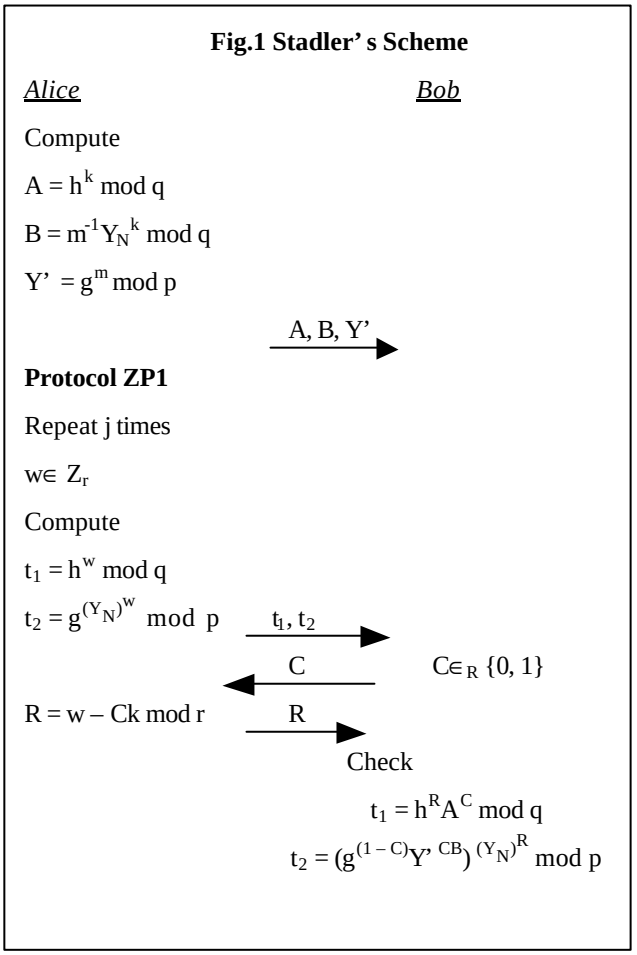
$$\text{and } t_2 = (g^{(1-C)} Y'^{\,CB})^{(Y_N)^R} \bmod p. \quad (4)$$

If they are correct, Bob accepts that the message has indeed been encrypted in the ciphertext (A, B).

To overview Stadler' s scheme, Fig.1 is given.

If Alice wants to send several encrypted messages to different recipients at the same time and convince Bob that those ciphertexts hind the corresponding messages. She must construct a single PVE scheme for each ciphertext with Bob, thus the overhead will be heavy. In next session, we will present our solution to reduce the overhead.

**Fig.1 Stadler' s Scheme**




**3. Publicly Verifiable Batch Encryption Scheme**

There are three roles in our scheme: Alice (the sender), Bob (the verifier), $U_i$ (a group of recipients). The private key/ public key pairs of Alice, Bob, and $U_i$ are ($X_A$, $Y_A = g^{X_A} \bmod p$), ($X_B$, $Y_B = h^{X_B} \bmod q$), ($X_i$, $Y_i = h^{X_i} \bmod q$), respectively.

Assume that Alice wants to send some messages to other participants ($U_i$) by using ElGamal encryption via

Bob. Bob must verify that those ciphertexts hind the correct messages without revealing messages. For example, assume that Alice wants to send the message $m_1$ to $U_1$, $m_2$ to $U_2$, .., $m_n$ to $U_n$. Alice must convince Bob that those messages have indeed been encrypted in the corresponding ciphertexts and can be decrypted by the recipients, respectively.

First Alice randomly chooses $k \in Z_r$ to compute $A = h^k \bmod q$. To avoid revealing $m_i$, she computes $B_i = m_i^{-1} Y_i^k \bmod q$ to encrypt $m_i$, where $i = 1, 2, .., n$ (The recipient can get the message by computing $m_i = A^{X_i} / B_i \bmod q$). She computes $Y_1' = g^{m_1} \bmod p$, $Y_2' = g^{m_2} \bmod p$, .., $Y_n' = g^{m_n} \bmod p$. Then she sends Bob $A, B_1, B_2, .., B_n, Y_1', Y_2', .., Y_n'$. Thus, Alice can convince Bob that the correct messages are sent to the recipients by running the following Protocol ZP2 j times.

**Protocol ZP2:**

Step 1: Alice randomly chooses $w \in Z_r$ to compute

$$t_1 = h^w \bmod q \quad (5)$$

$$\text{and } t_2 = g^{(Y_1^w + Y_2^w + .. + Y_n^w)} \bmod p. \quad (6)$$

Then she sends $t_1$ and $t_2$ to Bob.

Step 2: After receiving $t_1$ and $t_2$, Bob randomly chooses C and sends it to Alice, where C = 0 or 1.

Step 3: After receiving C, Alice computes $R = w - Ck \bmod r$ and sends R to Bob.

Step 4: After receiving R, Bob accepts that the message $m_i$ has indeed been encrypted in the ciphertext $(A, B_i)$ by checking whether

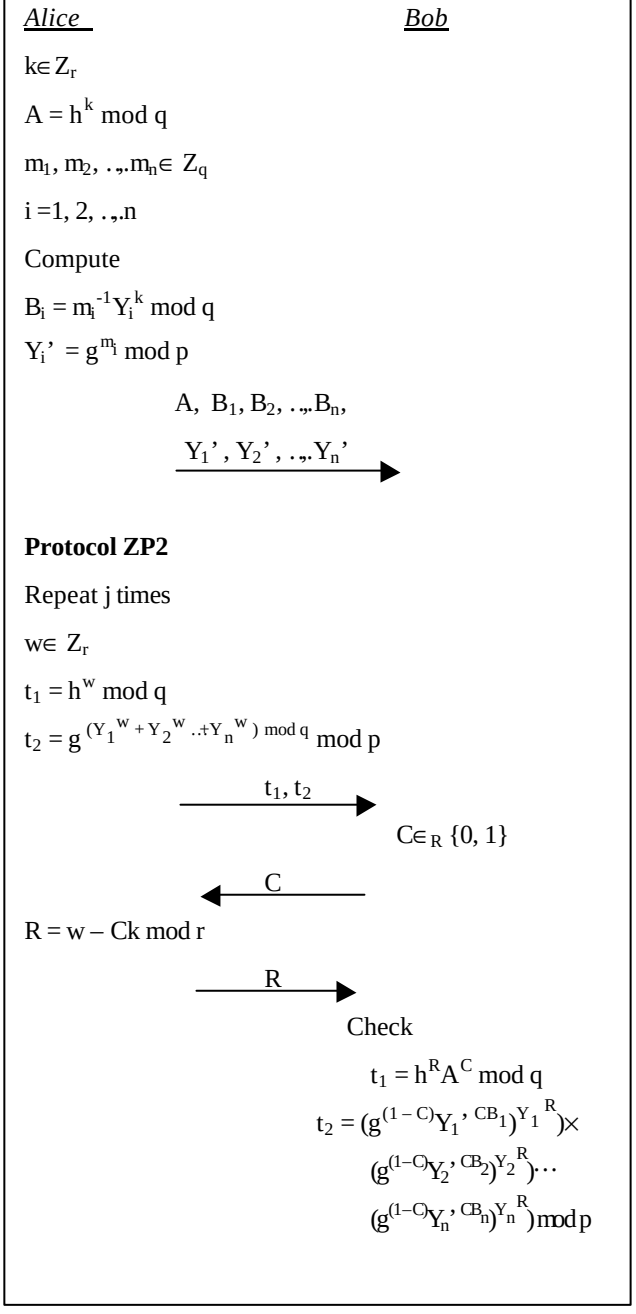
$$t_1 = h^R A^C \bmod q \quad (7)$$

$$\text{and } t_2 = (g^{(1-C)Y_1}, {}^{CB_1})^{Y_1 R}) (g^{(1-C)Y_2}, {}^{CB_2})^{Y_2 R}) \cdots (g^{(1-C)Y_n}, {}^{CB_n})^{Y_n R}) \bmod p. \quad (8)$$

To overview our PVBE scheme, Fig.2 is given.

We analyze the security and the efficiency of our scheme as follows: First, we used ElGamal encryption to encrypt messages. Given the pair $(A, B_i)$, the attacker cannot obtain the message $m_i$ without the recipient's secret key because he must face the discrete logarithm problem. Second, if one of the recipients tries to obtain other recipient's message from $Y_i'$, he also face the discrete logarithm problem. Furthermore, Protocol ZP2 is a zero-knowledge protocol similar to Stadler's Protocol ZP1 that the sender can successfully cheat with a probability of $2^{-j}$ at most.

**Fig.2 Public Verifiable Batch Encryption Scheme**




In the following, SPVEs denotes that ciphertexts are verified by running a single PVE scheme. We compare SPVEs with PVBE in Table 1. The comparison includes the number of modular exponentiations and multiplications, and the amount of transmitted data. In estimating the overhead of computations of n SPVEs and PVBE, mod p

and mod q are viewed as the same modular operation. Besides, modular additions or subtractions are neglected. Note that 1 modular inverse can be viewed as 1 modular exponentiation. 1 modular exponentiation requires on the average i|p| modular multiplications [6], where |p| denotes the length of the modular p and $0 < i \leq 1.5$. Fore example, i = 1.5 when the binary method is used.

In Stadler's scheme, Alice require 3 modular exponentiations, 1 modular multiplication and l modular inverse to compute A, B, and Y'. Thus, performing a single PVE scheme for n ciphertexts by running Protocol ZP1 j times, Alice and Bob require (3i|p|jn + 4i|p|n + n) and (3i|p|jn + jn) modular multiplications on the average, respectively.

In our scheme, Alice requires (2n + 1) modular exponentiations, n modular multiplications and n modular inverses to compute A, $B_1$, $B_2$, .., $B_n$, $Y_1'$, $Y_2'$, .., $Y_n'$. In Protocol ZP2, Alice requires (n + 2) modular exponentiations to compute $t_1$ and $t_2$. On the other hand, Bob must perform Equations (7) and (8). Equation (7) requires 1 modular exponentiation and 1 modular multiplication if C = 1, and 1 modular exponentiation if C = 0. However Equation (8) can be modified as follows:

$$t_2 = \begin{cases} g^{(Y_1{}^R + Y_2{}^R + .. + Y_n{}^R)} \bmod p & \text{if } C = 0 \quad (9) \\ (Y_1', {}^{B_1 Y_1{}^R})(Y_2', {}^{B_2 Y_2{}^R}) \\ \quad .. (Y_n', {}^{B_n Y_n{}^R}) \bmod p & \text{if } C = 1 \quad (10) \end{cases}$$

Performing Equation (9) requires (n + 1) modular exponentiations, while performing Equation (10) requires on the average n modular exponentiations and (n + ( |p| / w))modular multiplications at the cost of additional table by using the technique of the multi-exponentiation presented in [11]. Here w denotes the window size. Thus, performing our PVBE scheme for n ciphertexts by running Protocol ZP2 j times, Alice and Bob require ((3n + 1 + (n + 2)j)i|p| + n) and ((n + 1.5)ji|p| + ((n + 1 + |p| / w) / 2)j) modular multiplications on the average, respectively.

When j = 1, SPVEs requires on the average (10ni|p| + 2n) modular multiplications which is close to (10ni|p|) modular multiplications. Then, our scheme needs on the average ((5n + 4.5)i|p| + 1.5n + p/2w + 0.5) modular multiplications which is close to (5ni|p|) modular multiplications. Thus, our scheme reduces the overhead of computations by 66%. Next, we consider the case that j is large. On the average, SPVEs and our scheme requires approximately (6jni|p|) and (2jni|p|) modular multiplications, respectively. Thus, our scheme also reduces the overhead of computations by 66%.

**Table 1 Comparison between SPVEs and PVBE**

EXP = the number of modular exponentiations.

MUL = the number of modular multiplications.

TOTAL = EXP×i|p| + MUL.

| Various Scheme | | Computational cost | | |
|---|---|---|---|---|
| | | EXP | MUL | TOTAL |
| SPVE$_S$ | Sender | (4+3j)n | n | (4+3j)ni\|p\|+n |
| | Verifier | 3jn | jn | 3jni\|p\|+jn |
| PVBE | Sender | 3n+1+ ( n+2)j | n | ((3n+1+(n+2)j)i\|p\|+ n |
| | Verifier | (n+1.5)j | ((n+1+\|p\|/ w) /2)j | (n+1.5)ji\|p\|+ ((n+1+\|p\|/w)/2)j |

## 4. Applications

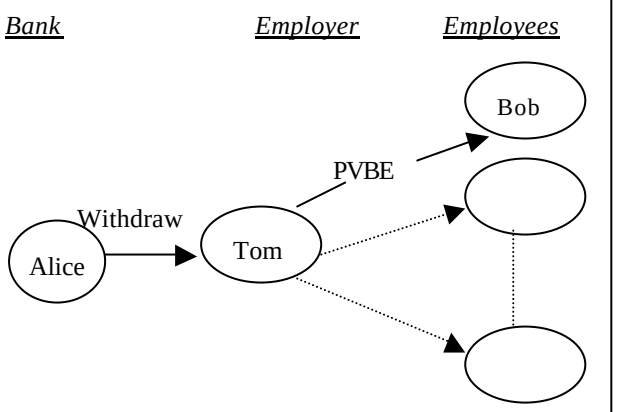
In [3, 9], they allow multiple coins can be withdrawn and spent using batch protocols, which are based on batch signatures schemes. The computing cost of withdrawal will be reduced for verifying large amount of coins.

In this section, we construct a batch escrowed e-cash system, which combines the publicly verifiable batch encryption and the batch signatures scheme.

In the batch escrowed e-cash system, there are four main participants: Alice (the bank), Tom (the employer), Bob (the escrowed agent of employees), and $U_i$ (employees), where i =1, 2, ..., n. The private key/public key pairs of Alice, Tom, Bob, and $U_i$ are ($X_A$, $Y_A = g^{X_A} \bmod p$), ($X_T$, $Y_T = h^{X_T} \bmod q$), ($X_B$, $Y_B = h^{X_B} \bmod q$), and ($X_i$, $Y_i = h^{X_i} \bmod q$), respectively. In a company (As illustration in Fig.3), Tom must withdraw the e-cash to pay employee's salary from the bank. Then, Tom distributes money to

every employee via Bob. Bob must check that Tom has indeed encrypted e-cash honestly.

**Fig.3 Escrowed Batch E-cash System**




First Tom sends Alice the request for withdrawal. Then Alice generates signature on a batch of n messages $m_1, m_2, \ldots, m_n$, where the message and the signature pair means a coin. Alice generates the signature as follows. First, Alice randomly chooses w' to compute $a = g^{w'} \bmod p$, $S = H_1(D_1||D_2||\ldots||D_n||a)$, and $R' = X_A S + w' \bmod q$, $D_i = H_0(m_i)$, $i = 1, 2, \ldots, n$, where $H_1(\cdot)$ and $H_0(\cdot)$ are secure hash functions. Then she sends Tom $m_1, m_2, \ldots, m_n, S, R', D_1, D_2, \ldots, D_n$.

After receiving $m_1, m_2, \ldots, m_n, S, R', D_1, D_2, \ldots, D_n$, Tom computes $a' = g^{R'} Y_A^{-S} \bmod q$ and check whether $H_0(m_i) = D_i$ and $S = H_1(D_1||D_2||\ldots||D_n||a')$ for verifying the signature $(S, R', D_1, D_2, \ldots, D_n, i)$ on message $m_i$. To overview the batch signatures scheme, Fig.4 is given.

After withdrawing the e-cash, Tom uses the PVBE scheme to dispense the e-cash to every employee. Tom computes the ciphertext pairs $(A, B_i)$ to encrypt the messages and the signature pair, and the public values $Y_i'$. Then he sends them to Bob by running Protocol ZP2 to convince Bob that Tom has indeed sent the e-cash to every employee.

**Fig.4 Batch Signatures Scheme**

| *Bank* | | *Tom* |
|---|---|---|
| $m_i \in Z_q, i = 1,2,\ldots,n$ | | |
| $w' \in Z_q$ | | |
| Compute | | |
| $a = g^{w'} \bmod p$ | | |
| $D_i = H_0(m_i)$ | | |
| $S = H_1(D_1\|\|D_2\|\|\ldots\|\|D_n\|\|a)$ | | |
| $R' = X_A S + w' \bmod q$ | | |
| | $m_1, m_2, \ldots, m_n, S,$ | |
| | $R', D_1, D_2, \ldots, D_n$ → | |
| | | Compute |
| | | $a' = h^{R'} Y_A^{-S} \bmod q$ |
| | | Check whether |
| | | $S = H_1(D_1\|\|D_2\|\|\ldots\|\|D_n\|\|a')$ |
| | | $H_0(m_i) = D_i$ |

**5. Conclusions**

In this paper, we had presented a publicly verifiable batch encryption scheme. It allows a third party to verify some encrypted messages without revealing any information. Compared to the SPVEs scheme, the PVBE scheme can on the average reduces the overhead of computations by 66%. Furthermore, The PVBE scheme can be applied to the escrowed e-cash system by using the batch protocol.

**Reference**

[1] N. Asokan, V. Shoup, and M. Waidner: "Optimistic fair exchange of digital signatures," *Advances in Cryptology—Proceeding of EUROCRYPT'98*, pp.591-606, 1998.

[2] G. Ateniese: "Efficient Verifiable Encryption (and Fair Exanchange) of Digital Signatures," *6th ACM Conference on Computer and Communication Security*, pp. 138-146, 1999.

[3] C. Boyd, E. Foo, and C. Pavloski: "Efficient Electronic Cash Using Batch Signatures," *4th Australasian*

*Conference on Information Security and Privacy, ACISP' 99*, pp.244-257, 1999.

[4] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch: "Verifiable secret sharing and achieving simultaneity in the presence of fault," *Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp.383-395, 1985.

[5] T. ElGamal: "A Public Key Cryptosystem and A Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, IT-31 (4), pp.469-472, 1985.

[6] D. M. Gordon: "A Survey of Fast Exponentiation Methods," *Journal of Algorithms*, 27(1), pp. 129-146,1998.

[7] P. W. Ko, J. E Hsien, and C. Y. Chen: "Escrowed Blind Signature and It's Application," *Proceedings of the Tenth National Conference on Information Security, ISC2000*, pp.119-122, 2000.

[8] W. Mao: "Verifiable Escrowed Signature," *Second Australian Conference in Information Security and Privacy*, pp.240-248, 1997.

[9] C. Pavloski, C. Boyd, and E. Foo: "Detachable Electronic Coins," *Proceeding of Second International Conference, Information and Communication Security, ICICS' 99*, pp.54-70, 1999.

[10] M. Stadler: "Publicly Verifiable Secret Sharing," *Advances in Cryptology — Proceeding of EUROCRYPT' 96*, pp.190-199, 1996.

[11] S. M. Yen, C. S. Laih and A. K. Lenstra: "Multi-exponentiation," *IEE Proc. Comput. Digit. Tech.*, Vol.141, No.6, pp.325-326, 1994.