Study Of A Range Proof Technique — Attack And Solution

Abstract. Peng and Bao propose a batch proof and verification technique and apply it to design efficient range proof with practical small ranges. Attacks are proposed in this paper against the batch proof and verification technique by Peng and Bao and its application to range proof. We show that their batch proof and verification fails in security such that a malicious prover without the claimed knowledge can pass the verification. As a result their range proof scheme to prove that a secret committed integer is in an interval range fails in soundness and cannot guarantee that the committed integer is in the range. To avoid the problem, we employ an efficient membership proof technique to replace the batch proof and verification technique in their range proof scheme and re-design it to achieve the claimed high efficiency with practical small ranges.

1 Introduction

Range proof is an applied cryptographic technique to enable a party to prove that a secret integer is in an interval range. The party chooses an integer from an interval range R, encrypts it or commits to it and publishes the ciphertext or commitment. Then he has to prove that the integer encrypted in the ciphertext or committed in the commitment is in R. The proof cannot reveal any information about the integer except that it is in the range. This proof operation is called range proof. The following security properties must be satisfied in a range proof protocol, while high efficiency is very important as well.

- Correctness: if the integer is in the range and the prover knows the integer and strictly follows the proof protocol, he can pass the verification in the protocol.
- Soundness: if the prover passes the verification in the protocol, the integer is guaranteed with an overwhelmingly large probability to be in the range.
- Privacy: no information about the integer is revealed in the proof except that it is in the range.

The most straightforward range proof technique is ZK (zero knowledge) proof of partial knowledge [7], which proves that the committed integer may be each integer in the range one by one and then link the multiple proofs with OR logic. It has a drawback: the number of computations it needs is linear to the size of the range, which leads to very low efficiency. The range proof schemes in [2, 9, 8] improve efficiency of range proof by discarding the cyclic group with public order in [7]. They notice that non-negativity of an integer x is binded in g^x when

the order of g is unknown. So in their range proofs they employ commitment of integers in Z instead of the traditional commitment of integers with a modulus. This special commitment function enables them to reduce a range proof in a range R to proof of non-negativity of integers. Although this method improves efficiency, it has two drawbacks. Firstly, its soundness depends on a computational assumption: when the multiplication modulus is a composite hard to factorize multiplication operation generates a large cyclic subgroup, whose order is secret and hard to calculate. So its soundness is only computational. Secondly, it cannot achieve perfect zero knowledge or simulatability. It reveals a statistically small (and intuitively-believed negligible) amount of information about the secret integer and only achieves the co-called statistical zero knowledge.

A special range proof scheme is proposed in [4]. On one hand it recognizes that sacrifice of unconditional soundness and perfect zero knowledge is necessary for high efficiency. On the other hand, it shows that asymptotical efficiency is not always the dominating factor in efficiency analysis. Actually, asymptotical efficiency in range proof is only important for large ranges. As the ranges in most practical applications of range proof are not large, an asymptotically higher cost may be actually lower in practice. So, the range proof scheme in [4] ignores asymptotical efficiency but focuses on the actual cost of range proof in practical small ranges. As a result, although its asymptotical efficiency is not the highest, it achieves the highest actual efficiency in practical small ranges and is more efficient in practical applications than the previous range proof solutions. However, [4] has its drawbacks as well. Besides conditional soundness like in [2, 9, 8], its has an additional limitation: its privacy depends on hardness of a special mathematical problem called ($\log_k$)-Strong Diffie Hellman assumption. Moreover, its efficiency advantage in practical small ranges is not great enough to dramatically improve efficiency of many applications.

The idea of range proof with actual high efficiency in practical small ranges is inherited by the most recent range proof scheme by Peng and Bao [10]. It overcomes the drawbacks of the existing range proof schemes like computational soundness, statistical privacy and additional assumption and is much more efficient than them when applied to practical small ranges. It is based on a new batch proof and verification technique extended from a batch proof and verification protocol in [6]. Batch verification is first proposed by Bellare *et al* [1] and then extended to batch proof and verification by Peng et al [11]. In [6], it is further extended to batch prove and verify multiple knowledge statements containing OR logic. The batch proof and verification protocol in [10] extends a batch proof and verification protocol in [6] and can prove and verify in a batch n instances of knowledge claims, each claiming knowledge of 1-out-of-k secret discrete logarithms. Peng and Bao claim that their batch proof and verification technique is secure and much more efficient than separately proving and verifying the n instances of knowledge claims. In their new range proof protocol, the committed integer is represented in a k-base coding system so that range proof in a range with width $b-a$ is reduced to n instances of range proof in Z_k where $b-a=k^n$. Then their batch proof and verification technique is employed to

batch the n instances of proof, so that efficiency of range proof in practical small ranges is greatly improved.

An attack is proposed in this paper to compromise security of the batch proof and verification technique by Peng and Bao. We show that their batch proof and verification fails in security such that a malicious prover without the claimed knowledge can pass the verification. Then it is illustrated that the attacking method can be applied to the range proof by Peng and Bao and break its soundness such that their range proof scheme cannot guarantee that the committed integer is in the range. To avoid the problem, we develop the pseudonym scheme by Brands *et al* [3] into an efficient membership proof, which is then employed to replace the batch proof and verification technique in the range proof scheme by Peng and Bao. The re-designed range proof technique can achieve the claimed high efficiency with practical small ranges in [10] without any compromise in security.

The rest of this paper is organized as follows. In Section 2, the batch proof and verification technique by Peng and Bao and its application to range proof are recalled. In Section 3, an attacking method is proposed to compromise security of their batch proof and verification technique. In Section 4, the attacking method is shown to be able to fail soundness of the range proof scheme by Peng and Bao. In Section 5, an efficient membership proof technique is employed to replace their batch proof and verification technique and re-design their range proof protocol to achieve their claimed efficiency without compromising security. The paper is concluded in Section 6.

2 The Batch Proof Technique and its Application to Range Proof by Peng and Bao

In [10], Peng and Bao propose a batch proof and verification protocol to batch prove and verify n knowledge statements, each of which claims knowledge of at least one of k discrete logarithms. Their proof protocol is recalled in Figure 1 where p, q are primes, $q|p-1$, G is the cyclic subgroup of Z_p^* with order q and g is a generator of G.

The batch proof and verification protocol in Figure 1 actually proves and verifies knowledge of $\log_g \prod_{i=1}^{n} y_{i,1}^{c_{i,1}}, \log_g \prod_{i=1}^{n} y_{i,2}^{c_{i,2}}, \ldots, \log_g \prod_{i=1}^{n} y_{i,k}^{c_{i,k}}$. It is claimed in [10] that its proof guarantees that with an overwhelmingly large probability the prover knows at least one of $\log_g y_{i,1}, \log_g y_{i,2}, \ldots, \log_g y_{i,k}$ for $i = 1, 2, \ldots, n$.

In [10], Peng and Bao employ the batch proof and verification protocol in Figure 1 to design a range proof scheme. In their design, a secret integer x is represented in a base-k coding system and then is efficiently proved to be in a range $\{a, a+1, \ldots, b\}$ where k is a parameter smaller than $b-a$. Their main idea is to reduce the range proof to $\log_k(b-a)$ instances of proof, which show that each digit of the base-k representation of $x-a$ is in Z_k. Then the $\log_k(b-a)$ instances of proof can be batched using the batch proof and verification technique in Figure 1 to improve efficiency.

Common input: (p, q, g), $\{(y_{i,1}, y_{i,2}, \ldots, y_{i,k})\}_{i=1,2,\ldots,n}$.
Knowledge to prove: $b_i \in \{1, 2, \ldots, k\}, v_{i,b_i}$ s.t. $y_{i,b_i} = g^{v_{i,b_i}} \bmod p$ for $i = 1, 2, \ldots, n$.
Denotation: $v_i = \{1, 2, \ldots, b_i - 1, b_i + 1, \ldots, k\}$.

1. The prover randomly selects $r_1, r_2, \ldots, r_k$ from Z_q and $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j \in v_i$ from Z_q. Then he computes

$$R_1 = g^{r_1} \prod_{1 \le i \le n,\ b_i = 1} \prod_{j \in v_i} y_{i,j}^{c_{i,j}} \bmod p$$
$$R_2 = g^{r_2} \prod_{1 \le i \le n,\ b_i = 2} \prod_{j \in v_i} y_{i,j}^{c_{i,j}} \bmod p$$
$$\ldots\ldots$$
$$\ldots\ldots$$
$$R_k = g^{r_k} \prod_{1 \le i \le n,\ b_i = k} \prod_{j \in v_i} y_{i,j}^{c_{i,j}} \bmod p$$
$$c_i = H(CI||c_{i-1}||c_{i-1,1}||c_{i-1,2}|| \ldots ||c_{i-1,k-1}) \text{ for } i = 1, 2, \ldots, n$$
$$c_{i,b_i} = c_i - \sum_{j \in v_i} c_{i,j} \bmod q \text{ for } i = 1, 2, \ldots, n$$
$$z_1 = r_1 - \sum_{\{i|b_i=1\}} c_{i,1} v_{i,1} \bmod q$$
$$z_2 = r_2 - \sum_{\{i|b_i=2\}} c_{i,2} v_{i,2} \bmod q$$
$$\ldots\ldots$$
$$\ldots\ldots$$
$$z_k = r_k - \sum_{\{i|b_i=k\}} c_{i,k} v_{i,k} \bmod q$$

   where CI is a bit string comprising common inputs in a certain order and

$$c_0 = R_1$$
$$c_{0,1} = R_2$$
$$c_{0,2} = R_3$$
$$\ldots\ldots$$
$$\ldots\ldots$$
$$c_{0,k-1} = R_k.$$

   It then sends
   $(z_1, z_2, \ldots, z_k, c_1, c_{1,1}, c_{1,2} \ldots, c_{1,k-1}, c_{2,1}, c_{2,2} \ldots, c_{2,k-1}, \ldots\ldots$
   $c_{n,1}, c_{n,2} \ldots, c_{n,k-1})$ to the verifier.

2. The verifier computes

$$c_{i,k} = c_i - \sum_{j=1}^{k-1} c_{i,j} \bmod q \text{ for } i = 1, 2, \ldots, n$$
$$c_i = H(CI||c_{i-1}||c_{i-1,1}||c_{i-1,2}|| \ldots ||c_{i-1,k-1}) \text{ for } i = 1, 2, \ldots, n$$

   and verifies

$$c_1 = H(CI||g^{z_1} \prod_{i=1}^{n} y_{i,1}^{c_{i,1}} \bmod p||g^{z_2} \prod_{i=1}^{n} y_{i,2}^{c_{i,2}} \bmod p$$
$$|| \ldots ||g^{z_k} \prod_{i=1}^{n} y_{i,k}^{c_{i,k}} \bmod p)$$

Fig. 1. Batch Proof and Verification of knowledge of 1-out-of-k Discrete Logarithms

In their range proof scheme, a secret integer x chosen from an interval range $\{a, a+1, \ldots, b\}$ is committed to in $c = g^x h^r \bmod p$ where h is a generator of G, $\log_g h$ is unknown and r is a random integer in Z_q. A party with knowledge of x and r has to prove that the message committed in c is in $\{a, a+1, \ldots, b\}$. The proof protocol and the corresponding verification are as follows.

1. $c' = c/g^a \bmod p$ and the proof that the integer committed in c is in $\{a, a+1, \ldots, b\}$ is reduced to proof that the integer committed in c' is in $\{0, 1, \ldots, b-a\}$.
2. The prover calculates representation of $x - a$ in the base-k coding system $(x_1, x_2, \ldots, x_n)$ to satisfy $x - a = \sum_{i=1}^{n} x_i k^{i-1}$ where for simplicity of description it is assumed $(b-a) = k^n$.
3. The prover randomly chooses $r_1, r_2, \ldots, r_n$ in Z_q and calculates and publishes $e_i = g^{x_i} h^{r_i} \bmod p$ for $i = 1, 2, \ldots, n$.
4. The prover publicly proves that he knows a secret integer $r' = \sum_{i=1}^{n} r_i k^{i-1} - r \bmod q$ such that $h^{r'} c' = \prod_{i=1}^{n} e_i^{k^{i-1}} \bmod p$ using zero knowledge proof of knowledge of discrete logarithm [12].
5. The range proof is reduced to n smaller-scale range proofs: the integer committed in e_i is in Z_k for $i = 1, 2, \ldots, n$. Those n instances of proof can be implemented through n instances of proof of knowledge of 1-out-of-k discrete logarithms

$$KN(\log_h e_i) \vee KN(\log_h e_i/g) \vee KN(\log_h e_i/g^2) \vee \ldots \\ \vee KN(\log_h e_i/g^{k-1}) \text{ for } i = 1, 2, \ldots, n \quad (1)$$

   where $KN(z)$ denotes knowledge of z.
6. Proof of (1) is implemented through batch proof and verification of knowledge of 1-out-of-k discrete logarithms in Figure 1.

It is claimed in [10] that this range proof technique is correct and sound. It achieves high efficiency when the range is not large and is suitable for applications needing to specify range proofs in practical small ranges.

3 Attack to Break the Batch Proof Protocol by Peng and Bao

In this section it is illustrated that with commitment $(y_{i,1}, y_{i,2}, \ldots, y_{i,k})$ for $i = 1, 2, \ldots, n$, $(R_1, R_2, \ldots, R_k)$ and hash-function-generated challenges $c_1, c_2, \ldots, c_n$ knowledge of $\log_g(R_j \prod_{i=1}^{n} y_{i,j}^{c_{i,j}})$ for $j = 1, 2, \ldots, k$ is not enough to guarantee knowledge of at least one integer in each set $\{\log_g y_{i,1}, \log_g y_{i,2}, \ldots, \log_g y_{i,k}\}$ for $i = 1, 2, \ldots, n$. We show how this proof mechanism can be attacked and broken. Firstly, a simple example of the attack is given as follows where $n = 3$ and $k = 2$, whose principle and effectiveness are proved in Theorem 1.

1. A malicious prover only knows $\log_g y_{2,2}$ and $\log_g y_{3,1}$.

2. The prover randomly chooses integers t_1, t_2 in Z_q and $y_{1,1}$, $y_{1,2}$ in G. Then he calculates $y_{2,1} = y_{1,1}^{t_1} \bmod p$ and $y_{3,2} = y_{1,2}^{t_2} \bmod p$.
3. The prover publishes $y_{1,1}, y_{1,2}, y_{2,1}, y_{2,2}, y_{3,1}, y_{3,2}$ and runs the proof protocol in Figure 1 in the case of $n = 3$ and $k = 2$.
4. In the proof protocol, the prover chooses v_1 and v_2 in Z_q and calculates $R_1 = g^{v_1} \bmod p$ and $R_2 = g^{v_2} \bmod p$.
5. The prover must provide c_1, c_2, c_3, $c_{1,1}$, $c_{1,2}$, $c_{2,1}$, $c_{2,2}$, $c_{3,1}$, $c_{3,2}$, $\log_g(R_1 y_{1,1}^{c_{1,1}} y_{2,1}^{c_{2,1}} y_{3,1}^{c_{3,1}})$ and $\log_g(R_2 y_{1,2}^{c_{1,2}} y_{2,2}^{c_{2,2}} y_{3,2}^{c_{3,2}})$ such that $c_1 = c_{1,1} + c_{1,2} \bmod q$, $c_2 = c_{2,1} + c_{2,2} \bmod q$, $c_3 = c_{3,1} + c_{3,2} \bmod q$, $c_1 = H(CI, R_1, R_2)$, $c_2 = H(CI, c_1, c_{1,1})$ and $c_3 = H(CI, c_2, c_{2,1})$ to pass the verification. This is feasible as illustrated in Theorem 1.

Theorem 1. *The malicious prover in the attack above does not need to know any* $\log_g y_{i,j}$ *other than* $\log_g y_{2,2}$ *and* $\log_g y_{3,1}$ *to pass the verification in Figure 1 when* $n = 3$ *and* $k = 2$.

Proof: As the prover knows $\log_g R_1$ and $\log_g y_{3,1}$ and his operations implies

$$\begin{aligned}\log_g(R_1 y_{1,1}^{c_{1,1}} y_{2,1}^{c_{2,1}} y_{3,1}^{c_{3,1}}) &= \log_g R_1 + \log_g(y_{1,1}^{c_{1,1}} (y_{1,1}^{t_1})^{c_{2,1}}) + \log_g y_{3,1}^{c_{3,1}} \\ &= \log_g R_1 + \log_g(y_{1,1}^{c_{1,1}} (y_{1,1}^{t_1})^{c_{2,1}}) + c_{3,1} \log_g y_{3,1} \\ &= \log_g R_1 + (c_{1,1} + t_1 c_{2,1}) \log_g y_{1,1} + c_{3,1} \log_g y_{3,1} \bmod q,\end{aligned}$$

the prover knows $\log_g(R_1 y_{1,1}^{c_{1,1}} y_{2,1}^{c_{2,1}} y_{3,1}^{c_{3,1}})$ if $c_{1,1} + t_1 c_{2,1} = 0 \bmod q$.

As the prover knows $\log_g R_1$, $\log_g y_1$ and $\log_g y_{2,1}$ and

$$\begin{aligned}\log_g(R_2 y_{1,2}^{c_{1,2}} y_{2,2}^{c_{2,2}} y_{3,2}^{c_{3,2}}) &= \log_g R_2 + \log_g y_{2,2}^{c_{2,2}} + \log_g(y_{1,2}^{c_{1,2}} (y_{1,2}^{t_2})^{c_{3,2}}) \\ &= \log_g R_2 + c_{2,2} \log_g y_{2,2} + \log_g(y_{1,2}^{c_{1,2}} (y_{1,2}^{t_2})^{c_{3,2}}) \\ &= \log_g R_2 + c_{2,2} \log_g y_{2,2} + +(c_{1,2} + t_1 c_{3,2}) \log_g y_{1,2} \bmod q,\end{aligned}$$

the prover knows $\log_g(R_2 y_{1,2}^{c_{1,2}} y_{2,2}^{c_{2,2}} y_{3,2}^{c_{3,2}})$ if $c_{1,2} + t_2 c_{3,2} = 0 \bmod q$.

So the prover can pass the verification if he can calculate $c_{1,1}$, $c_{1,2}$, $c_{2,1}$, $c_{2,2}$, $c_{3,1}$ and $c_{3,2}$ to satisfy

$$\begin{aligned} c_{1,1} + c_{1,2} &= c_1 \bmod q \\ c_{2,1} + c_{2,2} &= c_2 \bmod q \\ c_{3,1} + c_{3,2} &= c_3 \bmod q \\ c_{1,1} + t_1 c_{2,1} &= 0 \bmod q \\ c_{1,2} + t_2 c_{3,2} &= 0 \bmod q \\ c_1 &= H(CI||R_1||R_2) \\ c_2 &= H(CI||c_1||c_{1,1}) \\ c_3 &= H(CI||c_2||c_{2,1}) \end{aligned}$$

where t_1, t_2 are chosen by him. A solution as follows can calculate such $c_{1,1}$, $c_{1,2}$, $c_{2,1}$, $c_{2,2}$, $c_{3,1}$ and $c_{3,2}$ such that the attack can always succeed.

1. The prover calculates $c_1 = H(CI||R_1||R_2)$.
2. The prover randomly chooses $c_{1,1}$ in Z_q and calculates $c_{1,2} = c_1 - c_{1,1} \bmod q$.
3. The prover calculates $c_2 = H(CI||c_1||c_{1,1})$.
4. The prover calculates $c_{2,1} = (-c_{1,1})/t_1 \bmod q$ and $c_{2,2} = c_2 - c_{2,1} \bmod q$.
5. The prover calculates $c_3 = H(CI||c_2||c_{2,1})$.
6. The prover calculates $c_{3,2} = (-c_{1,2})/t_2 \bmod q$ and $c_{3,1} = c_3 - c_{3,2} \bmod q$.

□

Theorem 1 illustrates that a prover with knowledge of neither $\log_g y_{1,1}$ nor $\log_g y_{1,2}$ can always pass the verification in Figure 1 when $n = 3$ and $k = 2$. When n and k are larger, the attack is more flexible and a malicious prover has many concrete implementations to launch the attack and pass the verification in Figure 1 without the claimed knowledge. The following attacking algorithm is an example to break soundness of the batch proof and verification technique by Peng and Bao [10] as described in Figure 1 in general. It illustrates that a prover with knowledge of only $\log_g y_{n-1,1}$, $\log_g y_{n-1,2}$, $\ldots, \log_g y_{n-1,k-1}$, $\log_g y_{n,k}$ and no discrete logarithm of any other $y_{i,j}$ can always pass the verification in Figure 1. The attack is proved to be effective in Theorem 2.

1. A malicious prover only knows $\log_g y_{n-1,1}$, $\log_g y_{n-1,2}$, $\ldots, \log_g y_{n-1,k-1}$, $\log_g y_{n,k}$ and no discrete logarithm of any other $y_{i,j}$.
2. The prover randomly chooses integers $t_{i,1}, t_{i,2}, \ldots, t_{i,k}$ for $i = 2, 3, \ldots, n-2$ and $t_{n-1,k}, t_{n,1}, t_{n,2}, \ldots, t_{n,k-1}$ in Z_q. He randomly chooses $y_{1,1}, y_{1,2}, \ldots, y_{1,k}$ in G. Then he calculates

$$y_{i,j} = y_{1,j}^{t_{i,j}} \bmod p \text{ for } i = 2, 3, \ldots, n-2 \text{ and } j = 1, 2, \ldots, k$$
$$y_{n-1,k} = y_{1,k}^{t_{n-1,k}} \bmod p$$
$$y_{n,j} = y_{1,j}^{t_{n,j}} \bmod p \text{ for } j = 1, 2, \ldots, k-1.$$

3. The prover publishes $y_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$ and runs the proof protocol in Figure 1.
4. In the proof protocol, the prover chooses $v_1, v_2, \ldots, v_k$ in Z_q and calculates $R_j = g^{v_j} \bmod p$ for $j = 1, 2, \ldots, k$.
5. The prover must provide c_1, c_2, $\ldots, c_n$ and $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$ and $\log_g(R_j \prod_{i=1}^{n} y_{i,j}^{c_{i,j}})$ for $j = 1, 2, \ldots, k$ such that $c_i = \sum_{j=1}^{k} c_{i,j} \bmod q$ for $i = 1, 2, \ldots, n$, $c_i = H(CI||c_{i-1}||c_{i-1,1}||c_{i-1,2}||\ldots||c_{i-1,k-1})$ for $i = 1, 2, \ldots, n$, $c_0 = R_1$ and $c_{0,j} = R_{j+1}$ for $j = 1, 2, \ldots, k-1$ to pass the verification. This is feasible as illustrated in Theorem 2.

Theorem 2. *The malicious prover in the attack above does not need to know any* $\log_g y_{i,j}$ *for* $i = 1, 2, \ldots, n$ *and* $j = 1, 2, \ldots, k$ *other than* $\log_g y_{n-1,1}$, $\log_g y_{n-1,2}$, $\log_g y_{n-1,k-1}$, $\log_g y_{n,k}$ *to pass the verification in Figure 1.*

Proof: As the prover knows $\log_g R_j$, $\log_g y_{n-1,1}$, $\log_g y_{n-1,2}$, $\ldots, \log_g y_{n-1,k-1}$, $\log_g y_{n,k}$ and his operations implies

$$\begin{gathered}
\log_g(R_j \prod_{i=1}^{n} y_{i,j}^{c_{i,j}}) \\
= \log_g R_j + \log_g(y_{1,j}^{c_{1,j}} \prod_{i=2}^{n-2}(y_{1,j}^{t_{i,j}})^{c_{i,j}}) + \log_g y_{n-1,j} + \log_g(y_{1,j}^{t_{n,j}})^{c_{n,j}} \\
= \log_g R_j + c_{1,j} \log_g y_{1,j} + \log_g y_{1,j}^{\sum_{i=2}^{n-2} t_{i,j} c_{i,j}} + \log_g y_{n-1,j} + c_{n,j} t_{n,j} \log_g y_{1,j} \\
= \log_g R_j + (c_{1,j} + \sum_{i=2}^{n-2} t_{i,j} c_{i,j} + c_{n,j} t_{n,j}) \log_g y_{1,j} + c_{n-1,j} \log_g y_{n-1,j} \\
\bmod q \text{ for } j = 1, 2, \ldots, k-1
\end{gathered}$$

and

$$\begin{gathered}
\log_g(R_k \prod_{i=1}^{n} y_{i,k}^{c_{i,k}}) \\
= \log_g R_k + \log_g(y_{1,k}^{c_{1,k}} \prod_{i=2}^{n-1}(y_{1,k}^{t_{i,k}})^{c_{i,k}}) + \log_g y_{n,k}^{c_{n,k}} \\
= \log_g R_k + c_{1,k} \log_g y_{1,k} + \log_g y_{1,k}^{\sum_{i=2}^{n-1} t_{i,k} c_{i,k}} + \log_g y_{n,k}^{c_{n,k}} \\
= \log_g R_k + (c_{1,k} + \sum_{i=2}^{n-1} t_{i,k} c_{i,k}) \log_g y_{1,k} + c_{n,k} \log_g y_{n,k} \bmod q,
\end{gathered}$$

the prover knows $\log_g(R_j \prod_{i=1}^{n} y_{i,j}^{c_{i,j}})$ for $j = 1, 2, \ldots, k$ if

$$\begin{gathered}
c_{1,j} + \sum_{i=2}^{n-2} t_{i,j} c_{i,j} + c_{n,j} t_{n,j} = 0 \bmod q \text{ for } j = 1, 2, \ldots, k-1 \\
c_{1,k} + \sum_{i=2}^{n-1} t_{i,k} c_{i,k} = 0 \bmod q.
\end{gathered}$$

So the prover can pass the verification if he can calculate $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$ to satisfy

$$\begin{gathered}
\sum_{j=1}^{k} c_{1,j} = c_1 \bmod q \\
\sum_{j=1}^{k} c_{2,j} = c_2 \bmod q \\
\cdots \quad \cdots \\
\cdots \quad \cdots \\
\sum_{j=1}^{k} c_{n,j} = c_n \bmod q \\
c_{1,j} + \sum_{i=2}^{n-2} t_{i,j} c_{i,j} + c_{n,j} t_{n,j} = 0 \bmod q \text{ for } j = 1, 2, \ldots, k-1 \\
c_{1,k} + \sum_{i=2}^{n-1} t_{i,k} c_{i,k} = 0 \bmod q. \\
c_1 = H(CI || R_1 || R_2 || \ldots || R_k) \\
c_2 = H(CI || c_1 || c_{1,1} || c_{1,2}, \ldots, c_{1,k-1}) \\
\cdots \quad \cdots \\
\cdots \quad \cdots \\
c_n = H(CI || c_{n-1} || c_{n-1,1} || c_{n-1,2} || \ldots || c_{n-1,k-1})
\end{gathered}$$

where $t_{i,1}, t_{i,2}, \ldots, t_{i,k}$ for $i = 2, 3, \ldots, n-2$ and $t_{n-1,k}$, $t_{n,1}$, $t_{n,2}$, $\ldots, t_{n,k-1}$ are chosen by him. A solution as follows can calculate such $c_{i,j}$s and thus the attack can always succeed.

– The prover calculates $c_1 = H(CI||R_1||R_2||\ldots||R_k)$.
– The prover randomly chooses $c_{1,1}, c_{1,2}, \ldots, c_{1,k-1}$ in Z_q and calculates $c_{1,k} = c_1 - \sum_{j=1}^{k-1} c_{1,j} \bmod q$.
– The prover calculates $c_2 = H(CI||c_1||c_{1,1}||c_{1,2}, \ldots, c_{1,k-1})$.
– The prover randomly chooses $c_{2,1}, c_{2,2}, \ldots, c_{2,k-1}$ in Z_q and calculates $c_{2,k} = c_2 - \sum_{j=1}^{k-1} c_{2,j} \bmod q$.
– The prover calculates $c_3 = H(CI||c_2||c_{2,1}||c_{2,2}, \ldots, c_{2,k-1})$.
– The prover randomly chooses $c_{3,1}, c_{3,2}, \ldots, c_{3,k-1}$ in Z_q and calculates $c_{3,k} = c_1 - \sum_{j=1}^{k-1} c_{3,j} \bmod q$.
– ... ...
– ... ...
– The prover calculates $c_{n-2} = H(CI||c_{n-3}||c_{n-3,1}||c_{n-3,2}, \ldots, c_{n-3,k-1})$.
– The prover randomly chooses $c_{n-2,1}, c_{n-2,2}, \ldots, c_{n-2,k-1}$ in Z_q and calculates $c_{n-2,k} = c_{n-2} - \sum_{j=1}^{k-1} c_{n-2,j} \bmod q$.
– The prover calculates $c_{n-1} = H(CI||c_{n-2}||c_{n-2,1}||c_{n-2,2}, \ldots, c_{n-2,k-1})$.
– The prover calculates $c_{n-1,k} = (-c_{1,k} - \sum_{i=2}^{n-2} t_{i,k}c_{i,k})/t_{n-1,k} \bmod q$.
– The prover randomly chooses $c_{n-1,2}, c_{n-1,3}, \ldots, c_{n-1,k-1}$ in Z_q and calculates $c_{n-1,1} = c_{n-1} - \sum_{j=2}^{k} c_{n-1,j} \bmod q$.
– The prover calculates $c_n = H(CI||c_{n-1}||c_{n-1,1}||c_{n-1,2}||\ldots||c_{n-1,k-1})$.
– The prover calculates $c_{n,j} = (-c_{1,j} - \sum_{i=2}^{n-2} t_{i,j}c_{i,j})/t_{n,j} \bmod q$ for $j = 1, 2, \ldots, k-1$.
– The prover calculates $c_{n,k} = c_n - \sum_{j=1}^{k-1} c_{n,j} \bmod q$.

□

According to Theorem 2, to pass the verification in Figure 1 the prover only needs to know k instances of $\log_g y_{i,j}$ instead of one discrete logarithm in each of the n sets $\{\log_g y_{i,1}, \log_g y_{i,2}, \ldots, \log_g y_{i,k}\}$ for $i = 1, 2, \ldots, n$ as claimed in [10]. Actually, a malicious prover can pass the verification in Figure 1 using knowledge of any k instances of $\log_g y_{i,j}$ on the condition that they are not in the same set. Moreover, the malicious prover can choose more $y_{i,j}$ randomly than in our example of the attack. If he likes, he can even choose $y_{i,j}$ for $i = 1, 2, \ldots, n-2$ and $j = 1, 2, \ldots, k$ randomly in G and calculate another k instances of $y_{i,j}$ from more than one set as their functions to pass the verification on the condition that he knows discrete logarithm of the left k instances of $y_{i,j}$.

4 Attack to Break the Range Proof Scheme by Peng and Bao

In Section 3, it has been illustrated that the batch proof and verification protocol by Peng and Bao is not secure and cannot guarantee that the prover has the claimed knowledge. The attacking method proposed in Section 3 shows that in general applications of the batch proof and verification technique are vulnerable in security, as usually the $y_{i,j}$s are generated by the prover, otherwise he does not know the secret witness supposed to help him to pass the verification. However,

when applying the attacking method to the range proof scheme by Peng and Bao, we need to notice a special requirement in (1): $y_{i,j} = e_i/g^{j-1} \bmod p$ and thus $y_{i,j} = y_{i,j+1}g \bmod p$. So, as $\log_g h$ is secret, any one including the prover only knows at most one $\log_g y_{i,j}$ for each i in $\{1, 2, \ldots, n\}$. So the prover cannot know $\log_g y_{n-1,1}$, $\log_g y_{n-1,2}$, $\log_g y_{n-1,k-1}$ no matter whether he is malicious or not as assumed in the attacking algorithm in Section 3, which need to be adjusted to attack the range proof scheme as follow.

1. For simplicity of description, suppose a malicious prover only knows $\log_h e_{n-k+1}, \log_h e_{n-k+2}/g, \ldots, \log_h e_n/g^{k-1}$ instead of the knowledge statement in (1).
2. The malicious prover randomly chooses $e_1, e_2, \ldots, e_{n-k}$ from G.
3. He needs to choose $t_{i,j}$ for $i = 1, 2, \ldots, n-k$ and $j = n-k+1, n-k+2, \ldots, n$ to calculate $e_j = \prod_{i=1}^{n-k} e_i^{t_{i,j}} \bmod p$ for $j = n-k+1, n-k+2, \ldots, n$.
4. The prover publishes $y_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$ and runs the proof protocol in Figure 1.
5. In the proof protocol, the prover chooses $v_1, v_2, \ldots, v_k$ in Z_q and calculates $R_j = g^{v_j} \bmod p$ for $j = 1, 2, \ldots, k$.
6. The prover must provide c_1, c_2, $\ldots, c_n$ and $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$ and $\log_g(R_j \prod_{i=1}^{n} y_{i,j}^{c_{i,j}})$ for $j = 1, 2, \ldots, k$ such that $c_i = \sum_{j=1}^{k} c_{i,j} \bmod q$ for $i = 1, 2, \ldots, n$, $c_i = H(CI||c_{i-1}||c_{i-1,1}||c_{i-1,2}||\ldots||c_{i-1,k-1})$ for $i = 1, 2, \ldots, n$, $c_0 = R_1$ and $c_{0,j} = R_{j+1}$ for $j = 1, 2, \ldots, k-1$ to pass the verification where $y_{i,j} = e_i/g^{j-1} \bmod p$. This is feasible using the same method proposed in Theorem 1 and Theorem 2 due to a simple reason: although more equations are needed to satisfy in this attack against the range proof scheme to guarantee $y_{i,j} = e_i/g^{j-1} \bmod p$ we have $(n-k)k + nk$ integers (including $(n-k)k$ instances of $t_{i,j}$ and nk instances of $c_{i,j}$) to choose and only $k + k(k-1)$ linear equations to satisfy. According to the principle of linear algebra, we can always find $t_{i,j}$s and $c_{i,j}$s to pass the batch proof and verification and compromise soundness of the range proof scheme.

5 How to Avoid the Attack in Efficient Range Proof in Practical Small Ranges

Although the range proof scheme in [10] fails in soundness, its idea of committing to the secret integer in a k-base coding system when proving it to be in a small range is useful. We can base the idea on an efficient membership proof technique instead of the failed batch proof and verification protocol in [10]. We commit to the secret integer in n commitments, each of which contains an integer in Z_k. Then we can prove that each commitment really contains an integer in Z_k using membership proof.

5.1 An Efficient Membership Proof Technique

Membership proof is a cryptographic primitive to prove that a secret committed message m is in a finite set $S = \{s_1, s_2, \ldots, s_k\}$. We notice that in a pseudonym

scheme by Brands *et al* [3] a proof technique is proposed to prove that a secret message is not on a blacklist. The proof technique can prove that a committed secret integer is not a member of a set at a cost of $O(\sqrt{k})$ where k is the size of the set. This proof technique can be adapted to specify membership proof using the idea described as follows where for simplicity of description it is supposed that S can be divided into μ subsets $S_1, S_2, \ldots, S_\mu$ and each S_t contains ν integers $s_{t,1}, s_{t,2}, \ldots, s_{t,\nu}$.

1. For each S_t the ν-rank polynomial $F_t(x) = \prod_{i=1}^{\nu}(x - s_{t,i}) \bmod q$ is expanded into

   $$F_t(x) = \sum_{i=0}^{\nu} a_{t,i}x^i \bmod q$$

   to obtain the $\nu+1$ coefficients of the polynomial $a_{t,0}, a_{t,1}, \ldots, a_{t,\nu}$. Therefore, functions $F_t(x) = \sum_{i=0}^{\nu} a_{t,i}x^i$ for $t = 1, 2, \ldots, \mu$ are obtained, each to satisfy

   $$F_t(s_{t,i}) = 0 \text{ for } i = 1, 2, \ldots, \nu.$$

2. The prover calculates $c_i = c_{i-1}^m h^{r_i} \bmod p$ for $i = 2, 3, \ldots, \nu$ where $c_1 = c$ and r_i is randomly chosen from Z_q. The prover gives a zero knowledge proof that he knows m, r and r_i for $i = 2, 3, \ldots, \nu$ such that $c = g^m h^r \bmod p$ and $c_i = c_{i-1}^m h^{r_i} \bmod p$ for $i = 2, 3, \ldots, \nu$ using a simple combination of ZK proof of knowledge of discrete logarithm [12] and ZK proof of equality of discrete logarithms [5].
3. The prover proves that he knows $\log_h u_1$ or $\log_h u_2$ or ..... or $\log_h u_\mu$ using ZK proof of partial knowledge [7] where u_t can be publicly defined as

   $$u_t = \prod_{i=0}^{\nu} c_i^{a_{t,i}} \bmod p$$

   where $c_1 = c$ and $c_0 = g$. Actually the prover himself can calculate u_t more efficiently:

   $$u_t = \begin{cases} h^{\sum_{i=1}^{\nu} a_{t,i}R_i} \bmod p & \text{if } m \in S_t \\ g^{\sum_{i=0}^{\nu} a_{t,i}m^i} h^{\sum_{i=1}^{\nu} a_{t,i}R_i} \bmod p & \text{if } m \notin S_t \end{cases} \quad (2)$$

   where $R_i = mR_{i-1} + r_i \bmod q$ for $i = 2, 3, \ldots, \nu$ and $R_1 = r$.
4. Any verifier can publicly verify the prover's two zero knowledge proofs. He accepts the membership proof iff they are passed.

An interesting observation is that a verifier actually does not need to calculate

$$u_t = \prod_{i=0}^{\nu} c_i^{a_{t,i}} \bmod p \text{ for } t = 1, 2, \ldots, \mu$$

as it is costly. Instead, he only needs to verify validity of $u_1, u_2, \ldots, u_\mu$ calculated by the prover (through (2)) as follows.

1. He randomly chooses integers $\tau_1, \tau_2, \ldots, \tau_\mu$ from Z_q.

2. He verifies

$$\prod_{t=1}^{\mu} u_t^{\tau_t} = \prod_{i=0}^{\nu} c_i^{\sum_{t=1}^{\mu} \tau_t a_{t,i}} \bmod p,$$

which only costs $O(\mu + \nu)$ exponentiations.

This membership proof costs $O(\mu + \nu)$ in both computation (in terms of exponentiations) and communication (in terms of transfered integers). So it reduces the cost of general membership proof to $O(\sqrt{k})$ as $k = \mu\nu$. Its soundness is illustrated in Theorem 3.

Theorem 3. *The new membership proof is sound and the probability that the prover can pass its verification is negligible if* $m \neq s_i \bmod q$ *for* $i = 1, 2, \ldots, k$.

Proof: Suppose the prover commits to m in c where $m \neq s_i \bmod q$ for $i = 1, 2, \ldots, n$. If he passes the verification in the new membership proof with a non-negligible probability, it is guaranteed with a non-negligible probability that

$$c = g^m h^r \bmod p \tag{3}$$

$$c_i = c_{i-1}^m h^{r_i} \bmod p \text{ for } i = 2, 3, \ldots, \nu \tag{4}$$

where $c_1 = c$ and $c_0 = g$. As he passes the verification in the new membership proof with a non-negligible probability, there exists t in $\{1, 2, \ldots, \mu\}$ such that the prover knows R such that

$$h^R = \prod_{i=0}^{\nu} c_i^{a_{t,i}} \bmod p \tag{5}$$

with a non-negligible probability.

(3), (4) and (5) imply

$$h^R = g^{\sum_{i=0}^{\nu} a_{t,i} m^i} h^{\sum_{i=1}^{\nu} a_{t,i} R_i} \bmod p$$

where $R_i = mR_{i-1} + r_i \bmod q$ for $i = 2, 3, \ldots, \nu$ and $R_1 = r$. Namely

$$g^{\sum_{i=0}^{\nu} a_{t,i} m^i} h^{(\sum_{i=1}^{\nu} a_{t,i} R_i) - R} = 1 \bmod p.$$

So

$$\sum_{i=0}^{\nu} a_{t,i} m^i = 0 \bmod q \tag{6}$$

with a non-negligible probability as the employed commitment algorithm is binding and $g^0 h^0 = 1$. Note that $a_{t,0}, a_{t,1}, \ldots, a_{t,\nu}$ satisfy

$$\sum_{i=0}^{\nu} a_{t,i} s^{t,i} = 0 \bmod q \text{ for } i = 1, 2, \ldots, \nu. \tag{7}$$

So (6) and (7) imply

$$\begin{pmatrix} s_{t,1} & s_{t,1}^2 & \cdots & s_{t,1}^{\nu} \\ s_{t,2} & s_{t,2}^2 & \cdots & s_{t,2}^{\nu} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ s_{t,\nu} & s_{t,\nu}^2 & \cdots & s_{t,\nu}^{\nu} \\ m & m^2 & \ldots, & m^{\nu} \end{pmatrix} \begin{pmatrix} a_{t,1} \\ a_{t,2} \\ \cdots \\ \cdots \\ a_{t,\nu} \end{pmatrix} = \begin{pmatrix} -a_{t,0} \\ -a_{t,0} \\ \cdots \\ \cdots \\ -a_{t,0} \\ -a_{t,0} \end{pmatrix} \tag{8}$$

with a non-negligible probability. However, as $m \neq s_i \bmod q$ for $i = 1, 2, \ldots, k$ and all the calculations in the matrix is performed modulo q, $\begin{pmatrix} s_{t,1} & s_{t,1}^2 & \ldots & s_{t,1}^\nu \\ s_{t,2} & s_{t,2}^2 & \ldots & s_{t,2}^\nu \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ s_{t,\nu} & s_{t,\nu}^2 & \ldots & s_{t,\nu}^\nu \\ m & m^2 & \ldots, & m^\nu \end{pmatrix}$ is a non-singular matrix and thus (8) absolutely and always fails. Therefore, a contradiction is found and the probability that a prover can pass the verification in the new membership proof must be negligible if the integer he commits to in c is not in S. □

5.2 Range Proof Employing *k*-Base Coding and the Membership Proof

With the efficient membership proof present in Section 5.1, the k-base coding system in [10] can be inherited to design an efficient range proof protocol for practical small ranges as follows.

1. A party commits to a secret integer x in $c = g^x h^r \bmod p$ where h is a generator of G, $\log_g h$ is unknown and r is a random integer in Z_q. He then needs to prove that the message committed in c is in an interval range $\{a, a+1, \ldots, b\}$.
2. $c' = c/g^a \bmod p$ and the proof that the integer committed in c is in $\{a, a+1, \ldots, b\}$ is reduced to proof that the integer committed in c' is in $\{0, 1, \ldots, b-a\}$.
3. The prover calculates representation of $x - a$ in the base-k coding system $(x_1, x_2, \ldots, x_n)$ to satisfy $x - a = \sum_{i=1}^{n} x_i k^{i-1}$ where for simplicity of description it is assumed $(b-a) = k^n$.
4. The prover randomly chooses $r_1, r_2, \ldots, r_n$ in Z_q and calculates and publishes $e_i = g^{x_i} h^{r_i} \bmod p$ for $i = 1, 2, \ldots, n$.
5. The prover publicly proves that he knows a secret integer $r' = \sum_{i=1}^{n} r_i k^{i-1} - r \bmod q$ such that $h^{r'} c' = \prod_{i=1}^{n} e_i^{k^{i-1}} \bmod p$ using zero knowledge proof of knowledge of discrete logarithm [12].
6. The range proof is reduced to n smaller-scale membership proofs: the integer committed in e_i is in $\{0, 1, \ldots, k-1\}$ for $i = 1, 2, \ldots, n$. Those n instances of membership proof can be implemented through proof of

$$OPEN(e_i) = 0 \vee OPEN(e_i) = 1 \vee \ldots OPEN(e_i) = k - 1 \qquad (9)$$

   for $i = 1, 2, \ldots, n$ where $OPEN(z)$ denotes the opening to commitment z.
7. Proof of (9) is implemented through the efficient membership proof technique present in Section 5.1.

As explained in [10], such a design is very efficient for practical small ranges. For example, when $l = 10$, we can set $k = 4$ and $n = 2$ and it only costs $4\sqrt{k}n$

exponentiations and achieves high efficiency in practice. Correctness of the new range proof protocol is obvious and any reader can follow it step by step for verification. Its soundness depends on soundness of the employed membership proof protocol, which has been formally proved in Theorem 3. It is private as it employs standard zero knowledge proof primitives.

6 Conclusion

The batch proof and verification technique and its application to range proof by Peng and Bao [10] are compromised in security by the attacks proposed in this paper. Fortunately, after our modification, the range proof technique can still work securely and achieve the claimed high efficiency with practical small ranges.

References

1. M Bellare, J A Garay, and T Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250, Berlin, 1998. Springer-Verlag.
2. F Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Berlin, 2000. Springer-Verlag.
3. S Brands, L Demuynck, and B Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *ACISP '07*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415.
4. J Camenisch, R Chaabouni, and A Shelat. Efficient protocols for set membership and range proofs. In *ASIACRYPT '08*, volume 3089 of *Lecture Notes in Computer Science*, pages 234–252, Berlin, 2008. Springer-Verlag.
5. D Chaum and T Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Berlin, 1992. Springer-Verlag.
6. K Chida and G Yamamoto. Batch processing for proofs of partial knowledge and its applications. In *IEICE TRANS. FUNDAMENTALS, VOL.E91CA, NO.1 JANUARY 2008*, pages 150–159, 2008.
7. R Cramer, I Damgård, and B Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.
8. J Groth. Non-interactive zero-knowledge arguments for voting. In *ACNS '05*, volume 3531, pages 467–482, Berlin, 2005. Springer-Verlag. Lecture Notes in Computer Science.
9. H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *ASIACRYPT '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415, Berlin, 2003. Springer.
10. K Peng and F Bao. Batch range proof for practical small ranges. In *AFRICACRYPT 2010*, volume 6055 of *Lecture Notes in Computer Science*, pages 114–130, 2010.
11. K Peng and C Boyd. Batch zero knowledge proof and verification and its applications. In *ACM TISSEC Volume 10, Issue 2, Article No. 6 (May 2007)*, 2007.

12. C Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology, 4, 1991*, pages 161–174, 1991.