A Forward-Secure Blind Signature Scheme Based on the Strong RSA Assumption

Dang Nguyen Duc[1], Jung Hee Cheon[2], and Kwangjo Kim[1]

[1] International Research Center for Information Security (IRIS)
Information and Communication University (ICU)
58-4 Hwaam-dong, Yusong-gu, Deajeon, 305-732 Korea
{nguyenduc, kkj}@icu.ac.kr
http://www.iris.re.kr/

[2] School of Mathematical Science, Seoul National University (SNU)
San 56-1 Shillim-Dong, Kwanak-Gu, Seoul 151-747, Korea
jhcheon@math.snu.ac.kr

Abstract. Key exposures bring out very serious problems in security services. Especially, it is more severe in the applications such as electronic cash or electronic payment where money is directly involved. Forward secrecy is one of the security notions addressing the key exposure issues. Roughly speaking, forward secrecy is aimed to protect the validity of all actions using the secret key before the key exposure. In this paper, we investigate the key exposure problem in blind signature (with an application to the electronic cash in mind) and propose a blind signature scheme which guarantees forward secrecy. Our scheme is constructed from the provably secure Okamoto-Guillou-Quisquater (OGQ for short) blind signature scheme. Using the forking lemma by Pointcheval and Stern [4], we can show the equivalence between the existence of a forger with the solvability of the strong RSA problem. Further we show that our scheme introduces no significant communication overhead comparing with the original OGQ scheme.

1 Introduction

Digital signatures are the most well-known public key cryptography application which provides authentication of signing act. Clearly, the ability to sign (*i.e.*, owning the secret keys) must be available to the signer only. In practice, it is very difficult to guarantee that secret keys cannot be compromised since many implementation and administration errors can be exploited. To relax the problem, an intuitive solution is to use many secret keys - each valid only within a short period of time - and preferably keeps the public key unchanged over its lifetime. Such strategy is called *key evolution*.

However, key evolution must be designed carefully. For instance, if secret keys used in the past can be easily computed from the compromised secret key then key evolution does not help dealing with the key exposure problem. To address this issue, the notion of forward secrecy was introduced by Anderson [2]. Intuitively speaking, forward secrecy preserves security goal for all previous usage in case the current secret key is compromised. In other words, security goal is protected up to (*forward*) the time of secret key exposure.

An interesting extension of digital signature is blind signature proposed by Chaum [1]. Blind signature enables users to get a signer's signatures on their messages without revealing the message contents. Blind signature plays one of key ingredients in electronic cash system where the bank plays as the signer and customers play as users. Roughly, let's assume that a signature issued by the bank is equivalent to an electronic coin. Now, we consider the key exposure problem in case of blind signature (and so of an electronic cash system). It turns out that the key exposure problem in blind signature is very serious. Specifically, in electronic cash system, it is very severe since money is directly involved. When secret keys of the bank are stolen, attacker can generate as many valid electronic coins as he wants. Suppose that the bank is aware of key exposure and performs public key revocation. Since nobody can trust signature generated by using the stolen key, people who withdrawn their electronic coins but have not spent it, or who were paid electronic coins but have not deposited it will lose their money.

The first solution, the bank can think of, is to make stealing his secret keys essentially hard. For example, the bank can use secret sharing technique to distribute secret keys to several sites together with a threshold blind signature scheme to issue signatures. Clearly, this approach makes it more difficult for attackers to steal secret keys since they have to break in all sites holding shared secrets to learn the bank's secret keys. However, the above approach requires distributed computation that is very costly. Again, we turn to key evolution and forward secrecy. Specifically, the bank updates his secret key at discrete intervals and it is infeasible for an adversary to forge any signature valid in the past even if the current secret key is compromised. Blind signature is also seen to have other applications including electronic voting, auction, *etc.* All those applications are clearly vulnerable against key exposure problem. Thus relaxing the key exposure problem in blind signature is a useful feature not only in electronic cash but also in many other cryptographic applications.

Our approach to construct a forward secure blind signature scheme is to extend a well-studied blind signature scheme in the literature. We choose the Okamoto-Guillou-Quisquater (OGQ for short) blind signature scheme as our candidate. This scheme is constructed from the witness indistinguishable identification protocol based on Guillou-Quisquater identification protocol by Okamoto [8]. This blind signature scheme works on Z_N^* where N is a product of two large primes. The security of this scheme is proved by Pointcheval and Stern under random oracle model [4]. The scheme seems not to be vulnerable against generalized birthday attack [12] since this attack requires the knowledge of the order of the base group which is equivalent to factoring N.

In this paper, we present a forward secure blind signature scheme by extending the OGQ blind signature scheme. Our scheme exhibits an efficient key updating protocol and introduces no significant overhead comparing to the OGQ scheme.

The organization of the paper is as follows: In Section 2, we present background and definitions. The description of our forward secure blind signature scheme is given in Section 3. In Section 4, we analyze correctness, efficiency and security of our proposed scheme. Section 5 will be our conclusion and future work.

2 Background

2.1 The Key-evolving Blind Signature

In this section, we demonstrate a formal definition of a key-evolving blind signature scheme. The definition is adopted from the definition for a key-evolving digital signature given in [6].

Definition 1. *A key-evolving blind signature scheme consists of five algorithms,* `FBSIG` = <`FBSIG.Setup`, `FBSIG.Update`, `FBSIG.Signer`, `FBSIG.User`, `FBSIG.Verify`>, *where*

1. `FBSIG.Setup` *is a probabilistic polynomial-time algorithm which takes the security parameter k as its input and outputs system parameters including the initial secret key SK_1 and the public key PK of the signer.*
2. `FBSIG.Update` *is either deterministic or probabilistic algorithm. It takes the secret key SK_i for current time period, period i, as its input, and outputs a new secret key SK_{i+1} for time period $i + 1$.*
3. `FBSIG.Signer` *and* `FBSIG.User` *are a pair of probabilistic interactive Turing machines which model the signer and an user involving in a signature issuing session, respectively. Both machines have the following tapes: a read-only input tape, a write-only output tape, a read/write*

work tape, a read-only random tape and two communication tapes (one read-only and one write-only). The two machines may share a common read-only input tape as well. `FBSIG.Signer` *has its secret key* SK_i *on its input tape in time period* i*.* `FBSIG.User` *has a message* m *and the signer's public key* PK_i *on its input tape.* `FBSIG.Signer` *and* `FBSIG.User` *engage in a signature issuing protocol. After the protocol ends,* `FBSIG.Signer` *either outputs 'complete' or 'incomplete', and* `FBSIG.User` *either outputs signature of the message* m*,* $(i, \sigma(m))$*, or* $\perp$ *(i.e., error) respectively.*

4. `FBSIG.Verify` *is a deterministic algorithm which takes the public key of the signer,* PK*, and message, signature pair* $(m, i, \sigma(m))$ *as its input. It outputs either 'accept' or 'reject'. Clearly, for every valid signature,* `FBSIG.Verify` *must output 'accept'.*

We should emphasize that the period index, i, must be embedded into every signature. Otherwise, we cannot tell in which time period, the signature is issued.

2.2 Security Notions for a key-evolving Blind Signature with forward secrecy

Blindness. One characteristic of the ordinary cash is anonymity, meaning that user's buying activities can not be traced by the bank who issues cash. Blind signature clearly needs to address this issue since it is a means of cash issuance in electronic cash system. In fact, blindness is stronger than "obtaining signature without revealing message". To satisfies anonymity, blindness property implies that the signer cannot statistically distinguish signatures.

In a key-evolving blind signature, one may argue that since the time period index must be included in every signature. Then, the signer may use the time period index to uniquely identify every signature if he updates his secret keys after issuing each signature. So blindness property will be lost. However, the time period index j is publicly available and the signer must agree with all involved parties on when his secret keys should be updated. Another issue one may concern is that if a time period is too short, then there will be only a few signatures issued in that period. It may make the signer easier to identify signatures later on. This can be prevented by requiring a more rigorous blindness property. Let's consider the following game played by the signer (or any adversary that controls the signer) and two honest users, say U_0 and U_1.

- The signer chooses two messages m_0 and m_1.

- A referee chooses a random bit b and then m_b and m_{1-b} are given to U_0 and U_1, respectively.
- U_0 and U_1 engage with the signer to get signatures on their messages, m_b and m_{1-b}, respectively (not necessery in two different time periods since blindness property must be satisfied for all signatures, not just for signatures issued in one time period). Then, The two signatures are given to the signer. Finally, the signer outputs a guess for b, say b'. The signer wins the game if $b = b'$.

If probability that the signer wins the game is no better probability of guessing the random bit b given no information (*i.e.,* probability of $\frac{1}{2}$), the signer cannot link a signature to its owner. We say that blindness property is satisfied.

Forward Secrecy in Key-evolving Blind Signature. In different cryptographic schemes, forward secrecy may have different meanings depending on security goals for the schemes. In blind signature context, forward secrecy means unforgeability of signatures valid in previous time periods even if the current secret key of the signer is compromised.

2.3 Security Assumption

The security assumption of our scheme depends on the intractability of the strong RSA problem. The strong RSA problem is described as follows: Given a RSA modulus N (which is a product of two large primes) and a random element $c \in Z_N^*$, find m and $r \in Z_N^*$ such that $m^r = c \bmod N$. The strong RSA assumption implies that the strong RSA problem is intractable.

The strong RSA assumption is usually used with a special modulus N, *i.e.*, that is a product of two numbers, so called *safe primes.* We give definition of a safe prime as follows:

Definition 2. *Given a prime number q', if $q = 2q' + 1$ is also prime, we call q is a safe prime number. (q' is known as Sophie Germain prime.).*

3 Our Forward Secure Blind Signature Scheme

In this section, we describe our forward secure variant of the OGQ blind signature scheme. We denotes $\div$ by a division operation which gives the result as the quotient of the division (*i.e.*, if $a = qb + r$ then $a \div b = q$). The $\|$ denotes string concatenation. Also, we assume that a collision-free hash function H is available where its domain and codomain are $\{0,1\}^*$ and Z_λ^* (λ is a prime), respectively.

Firstly, we explain our idea on implementing a key-evolving protocol for the OGQ blind signature scheme. The OGQ scheme works on the multiplicative group Z_N^* where N is a product of two primes. Its secret key is a pair (r,s) and the corresponding public key is $V = a^{-r}s^{-\lambda}$ where a and λ are public (λ is also prime). Updating the secret s is easy, we just compute s' from s by squaring, say $s' = s^2$. However, updating r (in a way the new public key is related to the old public key) is difficult because we do not know the order of a in Z_N^*. If we compute V^2, we get $V^2 = a^{-2r}(s^2)^{-\lambda} \bmod N$. We cannot take $(2r, s^2)$ as a new secret key pair since it is trivially easy to get r from $2r$. To add randomness to the new r, we take a random exponent e from Z_N^* and compute $V^2a^e = a^{-2r+e}(s^2)^{-\lambda}$ mod N. l and r' denote the quotient and the remainder of $(2r-e)$ divided by λ, respectively. Then, we have $V^2a^e = a^{-r'}(a^l s^2)^{-\lambda} \bmod N$. Now, we can take V^2a^e as a new public key, $(r', s' = a^l s^2)$ as a new secret key. This key-evolving protocol is forward secure because in order to compute r or s from the new key pair (r', s') and $a^e \bmod N$, one needs to compute e from a^e or s from s^2. Since e is taken randomly, both of problems are very root finding problem in Z_N^*, which is equivalent to factoring N [14].

In an offline electronic cash system, payment can be made without online communication with the bank. In other words, verifiers should be able to verify signature without online communication with the signer. Therefore, in our case, a^e should be embedded into every signature so that verifier can compute the public key from V and the period index. One may argue that it is no better than generating the new key pair at random and including the public key into every signature. However, in blind signature, users are in charge of hashing their messages. Thus, users are under no obligation to embed the correct time period index into signatures (which means forward secrecy is lost). In contrast, the public key in our scheme is continuously squared after every period. So for verifiers to compute correct public key using period index (*i.e.*, V^{2^i}), users must embed the correct time period index into signatures.

We now describe each component of a five-tuple `FBSIG = <FBSIG.Setup`, `FBSIG.Update`, `FBSIG.Signer`, `FBSIG.User`, `FBSIG.Verify>`.

```
algorithm FBSIG.Setup($k$)
  Generate randomly two safe primes $p$ and $q$ of length k/2 bits
  $N \leftarrow pq$
  $\varphi(N) \leftarrow (q-1)(p-1)$
  Generate a random prime $\lambda$ such that it is co-prime with $\varphi(N)$
  Choose $a$ from $Z_N^*$ of order greater than $\lambda$
  Choose $r_0 \in_R Z_\lambda^*$ $s_0, e \in_R Z_N^*$
  $V \leftarrow a^{-r_0} s_0^{-\lambda} \bmod N$
  $f_1 \leftarrow a^e \bmod N$
  $v_1 \leftarrow V^2 a^e \bmod N$
  $l \leftarrow (2r_0 - e) \div \lambda$
  $r_1 \leftarrow (2r_0 - e) \bmod \lambda$
  $s_1 \leftarrow a^l s_0^2 \bmod N$
  Erase $p, q, e, r_0, s_0$ and $\varphi(N)$
  $SK_1 \leftarrow (1, r_1, s_1, v_1, f_1)$
  $PK \leftarrow (N, a, V, \lambda)$
  RETURN $(PK, SK_1)$

algorithm FBSIG.Update($SK_i$)
  $(i, r_i, s_i, v_i, f_i) \leftarrow SK_i$
  Choose $e \in_R Z_N^*$
  $v_{i+1} \leftarrow v_i^2 a^e \bmod N$
  $f_{i+1} \leftarrow f_i^2 a^e \bmod N$
  $l \leftarrow (2r_i - e) \div \lambda$
  $r_{i+1} \leftarrow (2r_i - e) \bmod \lambda$
  $s_{i+1} \leftarrow a^l s_i^2 \bmod N$
  $SK_{i+1} \leftarrow (i+1, r_{i+1}, s_{i+1}, v_{i+1}, f_{i+1})$
  Erase $SK_i, e$ and $l$
  RETURN $(SK_{i+1})$
```

Note that, i, v_i and f_i of SK_i are not secret anyway. We prefer to keep PK unchanged to avoid confusion because if public key is changed, we need to perform public key revocation. The signature issuing protocol is given as follows:

```
algorithm FBSIG.Signer($SK_i$)                algorithm FBSIG.User($PK, m$)

  On Error RETURN 'incomplete'                  On Error RETURN $\perp$

  $(i, N, \lambda, a, r_i, s_i, f_i) \leftarrow SK_i$
  Choose $t \in_R Z_\lambda^*$
  Choose $u \in_R Z_N^*$
  $x \leftarrow a^t u^\lambda \bmod N$
  Send $x$ to FBSIG.User
                                                Get $x$ from FBSIG.Signer
                                                $(N, \lambda, a, V) \leftarrow PK$
                                                Choose blinding factors $\alpha, \gamma \in_R Z_\lambda^*$ and $\beta \in_R Z_N^*$
                                                $x' \leftarrow x a^\alpha \beta^\lambda v_i^\gamma \bmod N$
                                                $c' \leftarrow H(i \parallel f_i \parallel m \parallel x')$
                                                $c \leftarrow (c' - \gamma) \bmod \lambda$
                                                Send $c$ to FBSIG.Signer
  Get $c$ from FBSIG.User
  $y \leftarrow (t + cr_i) \bmod \lambda$
  $w \leftarrow (t + cr_i) \div \lambda$
```

```
$z \leftarrow a^w u s_i^c \bmod N$
Send $y, z$ to FBSIG.User
                                        Get $y, z$ from FBSIG.Signer
                                        $y' \leftarrow (y + \alpha) \bmod \lambda$
                                        $w' \leftarrow (y + \alpha) \div \lambda$
                                        $w'' \leftarrow (c' - c) \div \lambda$
                                        $z' \leftarrow a^{w'} v_i^{-w''} z\beta \bmod N$
                                        $\sigma(m) \leftarrow (f_i, c', y', z')$

RETURN 'complete'                       RETURN $(i, \sigma(m))$
```

We assume that when users contact with the signer, i, v_i and f_i are available to users (*i.e.*, in the signer's read-only public directory). All users can access those information anonymously. The 'On Error' pseudo-code can be interpreted as 'Whenever an (unrecoverable) error occurs'. In practice, an error will be caused by a communication error between FBSIG.User and FBSIG.Signer.

To express the signature of a message, we will omit the index i on f_i since attackers (when try to forge a signature) do not have to use the correct f for a period).

```
algorithm FBSIG.Verify$(m, i, \sigma(m), PK)$
  $(N, \lambda, a, V) \leftarrow PK$
  $(f, c', y', z') \leftarrow \sigma(m)$
  $v_i \leftarrow V^{2^i} f \bmod N$
  $x'' \leftarrow a^{y'} z'^{\lambda} v_i^{c'} \bmod N$
  If $c' = H(i \parallel f \parallel m \parallel x'')$ then RETURN 'accept' else RETURN 'reject'
```

4 Analysis of FBSIG

4.1 Correctness

Theorem 1. *Suppose that* FBSIG.Signer *and* FBSIG.User *engage in a signature issuing protocol in period i such that* FBSIG.Signer *returns 'complete' and* FBSIG.User *returns signature on a message m, $(i, \sigma(m))$. Then,* FBSIG.Verify *always returns 'accept' on input $(PK, i, \sigma(m))$.*

Proof. We will show that $x'' = a^{y'} z'^{\lambda} (V^{2^i} f_i)^{c'} = x' \bmod N$. If the signature issuing protocol ends successfully then $f = f_i$ and we have:

$$a^{y'} z'^{\lambda} (V^{2^i} f_i)^{c'} = a^{y'} (a^{w'} v_i^{-w''} z\beta)^{\lambda} v_i^{c'} \bmod N$$

$$
\begin{aligned}
&= a^{y'} a^{w'\lambda} z^{\lambda} \beta^{\lambda} v_i^{c'-w''\lambda} \bmod N \\
&= a^{y'+w'\lambda} (a^{w} u s_i{}^{c})^{\lambda} \beta^{\lambda} v_i^{c'-w''\lambda} \bmod N \\
&= a^{y+\alpha} a^{w\lambda} u^{\lambda} s_i{}^{c\lambda} \beta^{\lambda} v_i^{c'-w''\lambda} \bmod N \\
&= a^{y+w\lambda} a^{\alpha} u^{\lambda} s_i{}^{c\lambda} \beta^{\lambda} v_i^{c'-w''\lambda} \bmod N \\
&= a^{t+cr_i} a^{\alpha} u^{\lambda} s_i{}^{c\lambda} \beta^{\lambda} v_i^{c'-w''\lambda} \bmod N \\
&= a^{t} u^{\lambda} a^{\alpha} (a^{-r_i} s_i^{-\lambda})^{-c} \beta^{\lambda} v_i^{c'-w''\lambda} \bmod N \\
&= x a^{\alpha} \beta^{\lambda} v_i^{-c} v_i^{c'-w''\lambda} \bmod N \\
&= x a^{\alpha} \beta^{\lambda} v_i^{(c'-c)-w''\lambda} \bmod N \\
&= x a^{\alpha} \beta^{\lambda} v_i^{\gamma} = x' \bmod N
\end{aligned}
$$

Hence $H(i \parallel f \parallel m \parallel x'') = H(i \parallel f \parallel m \parallel x') = c$ always holds which means that `FBSIG.Verify` always returns ‘accept’. □

4.2 Efficiency

We compare the key and signature sizes (in bits) of our key-evolving blind signature scheme and the OGQ blind signature scheme in the following table.

Scheme	Public Key Size	Secret Key Size	Signature Size
Our FBSIG	$5k + \log\lambda + \log(i)$	$k + \log\lambda$	$2k + 2\log\lambda + \log(i)$
OGQ Scheme	$3k + \log\lambda$	$k + \log\lambda$	$k + 2\log\lambda$

Note that $\log(i)$ is bit length of time period index. In terms of computational cost, the signature issuing procedure remains the same as the OGQ scheme. In verification process, we need to so some squaring operations to compute v_i. Our key updating is quite efficient. It needs three squaring operations, two exponentiations, one division and three multiplications in Z_N^*.

4.3 Security

Security of OGQ Blind Signature. In [4], the authors showed that one-more unforgeability is related to security of RSA cryptosystem. Even though the complexity of reduction step in their security proof is not polynomial in all security parameters, it is still one of the best result for blind signature.

We state two theorems regarding the security of our scheme as follows:

Theorem 2. *Our proposed scheme satisfies blindness property of a blind signature scheme.*

Proof. Let's consider the game played by an adversary $\mathcal{A}$ (the signer or the one controls the signer) and two honest users, U_0 and U_1 described in Section 2.2. If $\mathcal{A}$ receives $\perp$ from one of users, then he has no information to help guessing b other than a wild guess. Now suppose that he gets $(i, \sigma(m_b)) = (i, f_i, {c'}_b, {y'}_b, {z'}_b)$ and $(j, \sigma(m_{1-b})) = (j, f_j, {c'}_{1-b}, {y'}_{1-b}, {z'}_{1-b})$ from two users instead of $\perp$. Note that, what are exchanged between the signer and an user during signature issuing protocol are c, y and z. We call (c, y, z) is a view of the signer. We should show that, given any view (c, y, z) and any signature $(m, i, \sigma(m))$, there always exist uniquely blinding factors such that the resulting signature is $(m, i, \sigma(m))$ and the view of the signer is (c, y, z). This fact prevents the signer from deciding a given view corresponding to which signature since blinding factors are chosen randomly. The blinding factors α, β and γ can be uniquely computed given (c, y, z) and $(m, i, \sigma(m)) = (m, i, f, c', y', z')$ as follows: $\gamma = c' - c \bmod \lambda$, $\alpha = y' - y \bmod \lambda$ and $\beta = z'/(a^{w'} v_i^{-w''} z) \bmod N$ where w' and w'' are computed just like in the signature issuing protocol and $v_i = V^{2^i} f \bmod N$. To conclude, in any case, any adversary $\mathcal{A}$ cannot gain any helpful information during the signing protocol to guess b. In other words, his probability of success in guessing b is $1/2$. □

Theorem 3. *If there exists a forger which can break forward security of our scheme. Then, with non-negligible probability, we can violate the strong RSA assumption.*

Proof. A forger $\mathcal{F}$ obtains PK of the signer as its input, and interacts with the signer in an arbitrary way to get a set of message (of his choice) signature pairs MS. Whenever he wants, he breaks in the system (let say at time period b) and learns SK_b. Finally, with non-negligible probability, $\mathcal{F}$ outputs a forged message/signature pair for a time period $j < b$ which is not in the set MS. We need to simulate the signer to interact with $\mathcal{F}$ during signature issuing protocol and provide an hashing oracle to answer F's hashing queries. As usual, $\mathcal{F}$ can only interact with the signer polynomially many sessions and ask the hashing oracle polynomially many queries. We also need to provide a random tape for $\mathcal{F}$. First, we guess the period j that $\mathcal{F}$ will output a forged signature for that period. The break-in time of F must be period $b > j$. We can easily compute SK_b to answer $\mathcal{F}$'s break-in query by using the key setup and update procedure properly. We will run $\mathcal{F}$ twice with the same input PK. At the first time, assume that $\mathcal{F}$ outputs a forged signature $(j, \sigma_1(m)) = (j, f, c'_1, y'_1, z'_1)$ on a message m and the h-th query on the hashing oracle is $(j \parallel f \parallel m \parallel x'_1)$. It is expected that $V^{2^j} f = v_j \bmod N$. Otherwise, we retry from the beginning.

For the second time, we run $\mathcal{F}$ with the same random tape and answer to its hashing oracle queries the same values as in the first run until the h-th query, $(j \parallel f \parallel m \parallel x'_1)$. Due to the forking lemma [4], with non-negligible probability, $\mathcal{F}$ will again output a forged signature on message m for the period j, $(j, \sigma_1(m)) = (j, f, c'_2, y'_2, z'_2)$. Then it must be the case that $a^{y'_1} {z'_1}^{\lambda} (V^{2^j} f)^{c'_1} = a^{y'_2} {z'_2}^{\lambda} (V^{2^j} f)^{c'_2} \bmod N$. Thus, $a^{y'_1 - y'_2} (z'_1/z'_2)^{\lambda} = v_j^{e(c'_2 - c'_1)} \bmod N$ $(v_j = V^{2^j} f \bmod N)$. Since $v_j = a^{-r_j} {s_j}^{-\lambda} \bmod N$, we can come up with the following equation $a^{\rho} = b^{\lambda} \bmod N$ for some integer number ρ and b. This equation enables us to violate the strong RSA assumption due to the following lemma.

Lemma 1. *Given $a, b \in (Z/NZ)^*$, along with $\rho, \lambda \in Z$, such that $a^{\rho} = b^{\lambda}$ mod N and $gcd(\rho, \lambda) = 1$, one can efficiently compute $\mu \in Z_N^*$ such that $\mu^{\lambda} = a$ mod N.*

Proof. Since $gcd(\rho, \lambda) = 1$ we can use extended Euclidean algorithm to compute two integers ρ' and λ' such that $\rho\rho' = 1 + \lambda\lambda'$. Then, $\mu = b^{\rho'} a^{-\lambda'} \bmod N$ satisfies $\mu^{\lambda} = a \bmod N$.

Using the above lemma we can compute a λ-th root of a which contradicts with our security assumption, the RSA assumption since it is very likely that $gcd(\rho, \lambda) = 1$ (since λ is prime). □

5 Conclusions and Future Work

We presented the first forward secure blind signature scheme and analyzed its security. We believe that forward secrecy provides really useful features for a blind signature scheme, considering its applications such as electronic cash or electronic payment systems. Our scheme is as efficient as the original OGQ scheme. The key evolving protocol is efficient and supports unlimited time periods. However, the signature size of our scheme is two times of the original signature. Reducing the signature size is left as the future work.

Our scheme can also be extended to general groups whose orders are hard to find. In this case, the security assumption also changes to the strong root assumption [13] which is an analogy of the strong RSA assumption. An example of groups of unknown orders are class groups of imaginary quadratic orders. This generalization will be described in the full version of this paper.

Acknowledgment

The first author is grateful to Dr. Zhang Fangguo for his helpful discussion on blind signature. The second author was partially supported by SNU foundation in 2003.

References

1. David Chaum, *"Blind Signatures For Untraceable Payments"*, Advances in Cryptology - CRYPTO'82, Plenum Publishing, pp. 199-204, 1982.
2. Ross Anderson, *"Two Remarks on Public Key Cryptography"*, Invited Lecture, Fourth Annual Conference on Computer and Communications Security, ACM, 1997.
3. Louis S. Guillou and Jean J. Quisquater, *"A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing both Transmission and Memory"*, Advances in Cryptology - EUROCRYPT'88, LNCS 330, Springer-Verlag, pp. 123-128, 1988.
4. David Pointcheval and Jacques Stern, *"Provably Secure Blind Signatures Schemes"*, Advances in Cryptology - ASIACRYPT'96, LNCS 1163, Springer-Verlag, pp. 252-265, 1996.
5. Gene Itkis and Leonid Reyzin, *"Forward-Secure Signatures with Optimal Signing and Verifying"*, Advances in Cryptology - CRYPTO'01, LNCS 2139, Springer-Verlag, pp. 332-354, 2001.
6. Mihir Bellare and Sara K. Miner, *"A Forward-Secure Digital Signature Scheme"*, Advances in Cryptology - CRYPTO'99, LNCS 1666, Springer-Verlag, pp. 431-448, 1999.
7. Fangguo Zhang and Kwangjo Kim, *"ID-Based Blind Signature and Ring Signature from Pairings"*, Advances in Cryptology - ASIACRYPT'02, LNCS 2501, Springer-Verlag, pp. 533-547, 2002.
8. Tatsuki Okamoto, *"Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes"*, Advances in Cryptology - CRYPTO'92, LNCS 740, Springer-Verlag, pp. 31-53, 1992.
9. Ari Juels, Michael Luby and Rafail Ostrovsky, *"Security of Blind Signatures"*. Advanced in Cryptology - CRYPTO'97, LNCS 1294, Springer-Verlag, pp. 150-164, 1997.
10. Ronald Crammer and Victor Shoup, *"Signature Scheme Based on the Strong RSA Assumption"*, In ACM Transactions on Information and System Security, volume 3, pp. 161-185, 2000.
11. Claus P. Schnorr, *"Security of Blind Discrete Log Signatures Against Interactive Attacks"*, In Proceedings of ICISC'01, LNCS 2229, Springer-Verlag, pp. 1-12, 2001.
12. David Wagner, *"Generalized Birthday Problem"*, Advances in Cryptology - CRYPTO'02, LNCS 2442, Springer-Verlag, pp. 288-303, 2002.
13. Safuat Hamdy and Bodo Moller, *"Security of Cryptosystems Based on Class Groups of Imaginary Quadratic Orders"*, Advances in Cryptology - ASIACRYPT'00, LNCS 1976, Springer-Verlag, pp. 234-247, 2000.
14. Dan Boneh and Ramarathnam Venkatesan, *"Breaking RSA May Not Be Equivalent to Factoring"*, Advances in Cryptology - EUROCRYPT'98, LNCS 1403, Springer-Verlag, pp. 59-71, 1998.