Proxy Re-signature Schemes : Multi-use, Unidirectional & Translations

N.R.Sunitha
Department of Computer Science & Engg.
Siddaganga Institute of Technology,
Tumkur, Karnataka, India.
Email: nrsunitha@sit.ac.in

.

B.B.Amberker
Department of Computer Science & Engg.
National Institute of Technology,
Warangal, Andhra Pradesh, India.
Email: bba@nitw.ac.in

***Abstract*— In 1998, Blaze, Bleumer, and Strauss proposed proxy re-signatures, in which a semi-trusted proxy acts as a translator between Alice and Bob to translate a signature from Alice into a signature from Bob on the same message. Following are some open challenges in proxy re-signature schemes: i) the design of multi-use unidirectional systems and ii) determining whether or not a proxy re-signature scheme can be built that translates one type of signature scheme to another. We propose a solution for the first open problem using the property of forward-security. Our forward-secure proxy re-signature scheme which is based on the hardness of factoring translates one person's signature to another person's signature and additionally facilitates the signers as well as the proxy to guarantee the security of messages signed in the past even if their secret key is exposed today. To address the second open problem, we construct proxy signature schemes that translates Alice's Schnorr/ElGamal/RSA signature to Bob's RSA signature. The Signatures generated by regular signature generation algorithm and the proposed re-signature algorithms are indistinguishable.**

***Index Terms*— Signature translation, Proxy re-signature, Proxy Signature, Proxy revocation, Proxy key.**

I. Introduction

In Eurocrypt 98, the authors [1] proposed proxy re-signatures, in which a semi-trusted proxy acts as a translator between Alice and Bob. To translate, the proxy converts a signature from Alice into a signature from Bob on the same message. The proxy, however, does not learn any signing key and cannot sign arbitrary messages on behalf of either Alice or Bob. Since the BBS proposal, the proxy re-signature primitive has been largely ignored, until in [2] it was showed that it is a very useful tool for sharing web certificates, forming weak group signatures, and authenticating a network path.

The proxy signatures introduced in [3], [4] must not be confused with proxy re-signatures. A proxy signature [3], [4], [5], [6], [7] allows one user Alice, called the original signer, to delegate her signing capability to another user Bob, called the proxy signer. After that, the proxy signer Bob can sign messages on behalf of the original signer Alice. Upon receiving a proxy signature on some message, a verifier can validate its correctness by the given verification procedure. By this the verifier is convinced of the original signer's agreement on the signed message. In proxy re-signature, a proxy translates a perfectly-valid and publicly-verifiable signature, $\sigma_A(m)$, from Alice on a certain message, m , into a signature, $\sigma_B(m)$, from Bob on the same message m. Notice that, in proxy re-signature, the two signatures, one from Alice and the other from Bob as generated by the proxy, can coexist and both can be publicly verified as being two signatures from two distinct people on the same message. Moreover, the proxy can convert a single signature into multiple signatures of several and distinct signers, and vice-versa.

The authors in [2] re-opened the discussion of proxy re-signature by providing four separate results: (1) motivation for the need of improved schemes, by pointing out that the original BBS scheme [1], while satisfying their security notion, is unsuitable for most practical applications, including the ones proposed in the original paper, (2) formal definitions and a security model, (3) provably secure proxy re-signature constructions from bilinear maps, and (4) new applications. Nonetheless, they left open the following problems : i) Design of multi-use unidirectional scheme where the proxy is able to translate in only one direction and signatures can be re-translated several times. ii) Determining whether or not proxy re-signature scheme can be built that translate from one type of signature scheme to another i.e. like a scheme that translates Alice's Schnorr signatures into Bob's RSA based ones. In [8] the first constructions of multi-use unidirectional proxy re-signature wherein the proxy can only translate signatures in one direction and messages can be re-signed a polynomial number of times is discussed.

Further, Ateniese and Hohenberger, while formalising the primitive, pinned down the following useful properties that can be expected from proxy re-signature schemes.

doi:10.4304/jait.2.3.165-176

1. Unidirectional: re-signature keys can only be used for delegation in one direction.

2. Multi-use: a message can be re-signed a polynomial number of times.

3. Private Proxy: re-signature keys can be kept secret by an honest proxy.

4. Transparent: a user may not even know that a proxy exists.

5. Unlinkable: a re-signature cannot be linked to the one from which it was generated.

6. Key optimal: a user is only required to store a constant amount of secret data.

7. Non-interactive: the delegatee does not act in the delegation process.

8. Non-transitive: the proxy cannot re-delegate signing rights.

9. Temporary : revoke the rights given to proxy.

The construction given by Blaze et al. is bidirectional and multi-use. However, in [2] authors have pinpointed a flaw in the latter scheme: given a signature/re-signature pair, anyone can deduce the re-signature key that has been used in the delegation (i.e. the private proxy property is not satisfied). Another issue in [1] is that the proxy and the delegatee can collude to expose the delegators secret. To overcome these limitations, Ateniese and Hohenberger proposed two constructions based on bilinear maps. The first one is a multi-use, bidirectional protocol built on BLS signatures [9]. Their second scheme is unidirectional (the design of such a scheme was an open problem raised in [1]) but single-use. It involves two different signature algorithms: first-level signatures can be translated by the proxy whilst second-level signatures cannot. A slightly less efficient variant was also suggested to ensure the privacy of re-signature keys kept at the proxy. The security of all schemes was analyzed in the random oracle model [10]. In [11], an identity based proxy re-signature scheme is proposed that is existentially unforgeable in the standard model under the Strong Diffie-Hellman assumption. In [12], the authors discuss the design of generic unidirectional proxy re-signature scheme using homomorphic signatures and incorporation of forward-security into the proxy re-signature paradigm. In [13], the authors have constructed a blind proxy re-signature scheme. In [14], a forward-secure threshold proxy re-signature scheme in the standard model is proposed. In [15], the authors propose a multi-use bidirectional ID based proxy re-signature scheme.

Digital signatures are vulnerable to leakage of secret key. If the secret key is compromised, any message can be forged. To prevent future forgery of signatures, both public key and secret key must be changed. Notice, that this will not protect previously signed messages: such messages will have to be re-signed with new pair of public key and secret key, but this is not feasible. Also changing the keys frequently is not a practical solution. To address the above problem, the notion of forward security for digital signatures was first proposed in [16], and carefully formalised in [17] (see also[18], [19], [20], [21]). The basic idea is to extend a standard digital signature scheme with a key updation algorithm so that the secret key can be changed frequently while the public key stays the same.

To address the first open problem, we propose a new construction for multi-use (i.e. the translation of signatures can be performed in sequence and multiple times by distinct proxies) unidirectional (i.e. the proxy information allows translating signatures in only one direction) proxy re-signature scheme using the property of forward-security. Our forward-secure proxy re-signature scheme, based on the hardness of factoring, translates one persons signature to another persons signature and additionally facilitates the signers as well as the proxy to guarantee the security of messages signed in the past even if their secret key is exposed today (property of forward-security). With a minor change in resigning key, we can make the scheme to behave as a multi-use bidirectional scheme. The scheme also satisfies the properties viz. private proxy, transparent, unlinkable, key optimal, interactive, non-transitive and temporary.

To address the second open problem of Ateniese and Hohenberger, we present construction of schemes which converts Alice's Schnorr/ElGamal signature to Bob's RSA signature. We construct this by generating suitable proxy re-sign keys which are computed by establishing communication among delegatee, proxy signer and the delegator. At no point of conversion the security of Schnorr, ElGamal and RSA signature schemes are compromised. Signatures of Schnorr and ElGamal get converted to RSA signatures by providing only the signature and re-sign keys as input to Re-sign algorithm.

The organisation of our paper is as follows: In Section 2, we define the proxy re-signature, explain two of our schemes i.e Forward-Secure Bi-directional Multi-use Proxy Re-Signature Scheme and Forward-Secure Unidirectional Multi-use Proxy Re-Signature Scheme and also discuss the application of proxy re-signatures in banking environment. In Section 3, we explain three conversion schemes i.e Schnorr to RSA conversion Scheme ElGamal to RSA conversion Scheme and RSA to RSA conversion scheme and provide a comparison among various proxy re-signature schemes. In Section 4, we discuss properties and security of the proposed schemes. In Section 5, proxy revocation is explained. Lastly in Section 6, we conclude.

II. Forward-Secure Proxy Re-signature Scheme

As digital signatures, proxy re-signatures are also vulnerable to leakage of re-signing key. If the re-signing key is compromised, any one can become a proxy. To prevent future forgery of re-signatures, both the delegator as well as the delegatee must change their public key and secret key pair and a new re-signing key computed. But this will not protect previously signed messages: such messages will have to be re-signed with new pair of public

key and secret key which is not feasible. To address this problem, we use the concept of forward security for proxy re-signatures.

To translate Alice's signature to Bob's signature, the secret and public keys are generated as indicated in the Key Generation algorithm. We know that a forward secure signature scheme has its operation divided into time periods, each of which uses a different secret key to sign a message. The new secret keys are generated as described in the Key Evolution algorithm. Alice signs in any time period j using the Signature Generation algorithm. This signature is required to be converted to Bob's signature. Using the protocol indicated in the Re-Signature Key Generation the proxy generates the re-signature key $rk_{A\to B}$ and executes the Re-sign algorithm to translate Alice's signature to Bob's signature. This scheme works for a period of T time periods, i.e. the proxy has the power to resign only for T time periods and after the expiry of T time periods the proxy is automatically revoked.

A. Definition of Proxy Re-signature scheme

We follow the definitions given in [2]. A proxy re-signature scheme is a tuple of polynomial time algorithms (KeyGen, ReKey, Sign, ReSign, Verify), where, (KeyGen, Sign, Verify) form the standard key generation, signing, and verification algorithms.

On input $(pk_A, sk_A^*, pk_B, sk_B)$, where (pk_A, sk_A^*) is the public key - secret key pair of A and (pk_B, sk_B) is the public key - secret key pair of B, the re-signing key generation algorithm, ReKey, outputs a key $rk_{A\to B}$ for the proxy. By providing this key as input to ReSign algorithm, the proxy converts a signature from Alice into a signature of Bob on the same message. The proxy, however, does not learn any signing key and cannot sign arbitrary messages on behalf of either Alice or Bob. (Note: $rk_{A\to B}$ allows to transform *A's* signatures into *B's* signatures, thus B is the delegator and A is the delegatee). The input marked with a * is optional as in case of public proxy re-signature scheme.
On input $rk_{A\to B}$, a public key pk_A, a signature $\sigma_A(m)$, and a message m, the re-signature function, ReSign, outputs $\sigma_B(m)$ if Verify$(pk_A, m, \sigma_A(m)) = 1$ and an error message otherwise.

The correctness of the scheme has two requirements. For any message m in the message space and any key pairs $(pk, sk), (pk', sk') \leftarrow KeyGen(1^k)$, let $\sigma = Sign(sk, m)$ and $rk \leftarrow ReKey(pk, sk, pk', sk')$. Then the following two conditions must hold:

$$Verify(pk, m, \sigma) = 1$$

$$Verify(pk', m, ReSign(rk, \sigma)) = 1.$$

That is, all signatures formed by either the signing or re-signing algorithms will pass verification.

B. Multi-use Bi-directional Proxy Re-Signature Scheme

We propose a new construction for multi-use bidirectional proxy re-signature scheme using the property of forward-security. Our forward-secure proxy re-signature scheme, based on the hardness of factoring, translates one persons signature to another persons signature and additionally facilitates the signers as well as the proxy to guarantee the security of messages signed in the past even if their secret key is exposed today (property of forward-security). With a minor change in resigning key, we can make the scheme to behave as a multi-use bidirectional scheme. The scheme also satisfies the properties viz. private proxy, transparent, unlinkable, key optimal, interactive, non-transitive and temporary. Following are the algorithms for the Forward-Secure Multi-use Bi-directional Proxy Re-Signature Scheme.

1) **Key generation:** Both Alice and Bob generate the keys by running the following algorithm which takes as input the security parameter k, the number l of points in the keys and the number T of time periods over which the scheme is to operate. The notations are same as in Bellare-Miner Forward-secure signature scheme discussed in Chapter 3. p, q are random distinct $k/2$ bit primes each congruent to $3 \bmod 4$. $N \leftarrow p.q$. Alice and Bob agree upon common N.
   Alice's keys: The base secret key $SKA_0 = (SA_{1,0}, \ldots, SA_{l,0}, N, 0)$ (where $SA_{i,0} \overset{R}{\leftarrow} Z_N^*$ and N is a Blum-Williams integer). For verifying signatures the verifier is given the public key PKA, calculated as the value obtained on updating the base secret key $T+1$ times: $PKA = (UA_1, \ldots, UA_l, N, T)$ where $UA_i = SA_{i,0}^{2^{T+1}} \bmod N_A, i = 1, \ldots, l$.
   Bob's keys: The base secret key $SKB_0 = (SB_{1,0}, \ldots, SB_{l,0}, N, 0)$ (where $SB_{i,0} \overset{R}{\leftarrow} Z_N^*$ and N is a Blum-Williams integer). For verifying signatures the verifier is given the public key: $PKB = (UB_1, \ldots, UB_l, N, T)$ where $UB_i = SB_{i,0}^{2^{T+1}} \bmod N_B, i = 1, \ldots, l$.
2) **Key evolution:** During time period j the signer signs using key SK_j. This key is generated at the start of period j by applying a key update algorithm to the key SK_{j-1}. The update algorithm squares the l points of the secret key at the previous stage to get the secret key at the next stage.
   Key evolution for Alice: The secret key $SKA_j = (SA_{1,j}, \ldots, SA_{l,j}, N_A, j)$ of the time period j is obtained from the secret key $SKA_{j-1} = (SA_{1,j-1}, \ldots,$
   $SA_{l,j-1}, N_A, j-1)$ of the previous time period via the update rule: $SA_{i,j} = SA_{i,j-1}^2 \bmod N_A, i = 1, \ldots, l; j = 1, \ldots, T$.
   Key evolution for Bob: The secret key $SKB_j = (SB_{1,j}, \ldots, SB_{l,j}, N_B, j)$ of the time period j is obtained from the secret key $SKB_{j-1} = (SB_{1,j-1}, \ldots,$

$SB_{l,j-1}, N_B, j-1)$ of the previous time period via the update rule: $SB_{i,j} = SB_{i,j-1}^2 \bmod N_B, i = 1,\ldots,l; j = 1,\ldots,T$.

3) **Re-Signature Key Generation (ReKey):** On input two secret keys $SKA_j = (SA_{1,j},\ldots,SA_{l,j}, N_A, j)$ and $SKB_j = (SB_{1,j},\ldots,SB_{l,j}, N_b, j)$, the re-signature key, $rk_{A\to B,j} = (rk_{1,j},\ldots,rk_{l,j})$ is computed as

$$rk_{i,j} = SB_{i,j}/SA_{i,j} \bmod N$$

where $i = 1,\ldots,l; j = 1,\ldots,T$. The scheme can be used as a bidirectional multi-use proxy re-signature scheme.
Observe that the key $rk_{A\to B}$ can be securely generated as follows:

a) Proxy sends a random $r \in Z_N^*$ to Alice.
b) Alice sends $(r/SA_{1,j},\ldots,r/SA_{l,j})$ to Bob.
c) Bob sends $(r(SB_{1,j}/SA_{1,j}),\ldots,r(SB_{l,j}/SA_{l,j})$ to the proxy.
d) Proxy recovers $(SB_{1,j}/SA_{1,j},\ldots,SB_{l,j}/SA_{l,j})$.

Key evolution for Proxy: The re-signature key $rk_{A\to B,j} = (rk_{1,j},\ldots,rk_{l,j})$ of the time period j is obtained from the re-signature key $rk_{A\to B,j-1} = (rk_{1,j-1},\ldots,rk_{l,j-1})$ of the previous time period via the update rule: $rk_{i,j} = rk_{i,j-1}^2 \bmod N, i = 1,\ldots,l; j = 1,\ldots,T$.

4) **Signature Generation:** It has as input the secret key SKA of the current period, the message M to be signed, and the value j of the period itself to return a signature $\langle j,(Y,Z)\rangle$ where Y, Z in Z_N^* are calculated as follows:

$$Y = R^{2^{(T+1-j)}} \bmod N, \quad \text{where } R \overset{R}{\leftarrow} Z_N^* \quad (1)$$

$$Z = R\prod_{i=1}^{l} SA_{i,j}^{c_i} \bmod N, \quad (2)$$

$c_1,\ldots,c_l = H(j,Y,M)$ being the l output bits of a public hash function.

5) **Re-Sign (ReSign):** We verify the signature before we re-sign. On input a re-signature key $rk_{A\to B,j}$, a public key PKA, a signature $\langle j,(Y,Z)\rangle$, and a message M, we check that Verify$(PKA, m, \langle j,(Y,Z)\rangle)$ = 1. If the signature,$\langle j,(Y,Z)\rangle$, does not verify, re-signing is not done and an error message is displayed.
If the signature is verified, we set

$$Z' = Z\prod_{i=1}^{l} rk_{i,j}^{c_i} \bmod N,$$

where $c_1,\ldots,c_l = H(j,Y,M)$ and output the signature $\langle j,(Y,Z')\rangle$. Observe that

$$\begin{aligned} Z' &= Z\prod_{i=1}^{l} rk_{i,j}^{c_i} \bmod N \\ &= R\prod_{i=1}^{l} SA_{i,j}^{c_i}(\prod_{i=1}^{l} SB_{i,j})^{c_i}/(\prod_{i=1}^{l} SA_{i,j})^{c_i} \\ &\quad \bmod N \\ &= R\prod_{i=1}^{l} SB_{i,j}^{c_i}, \end{aligned}$$

which shows that the signature $\langle j,(Y,Z')\rangle$ is Bob's signature. Thus, Re-Sign has translated Alice's signature into Bob's signature.
Though, just as in BBS scheme, our scheme also computes the resigning key as the ratio of secret keys of Alice and Bob, but the resigning key cannot be computed using the signature/re-signature pair. BBS proxy re-signature scheme is briefly described in Appendix A.9. Let the re-signature key be

$$rk_{A\to B,j} = (rk_{1,j},\ldots,rk_{l,j})$$

where $rk_{i,j} = SB_{i,j}/SA_{i,j} \bmod N$. Let $\langle j,(Y,Z)\rangle$ and $\langle j,(Y,Z')\rangle$ be the signature and re-signature pair respectively, where $Z = R\prod_{i=1}^{l} SA_{i,j}^{c_i} \bmod N$ and $Z' = R\prod_{i=1}^{l} SB_{i,j}^{c_i} \bmod N$. The ratio of re-signature to signature is

$$\begin{aligned} Z'/Z &= (R\prod_{i=1}^{l} SB_{i,j}^{c_i} \bmod N)/ \\ &\quad (R\prod_{i=1}^{l} SA_{i,j}^{c_i} \bmod N) \\ &= (\prod_{i=1}^{l} SB_{i,j}^{c_i}/\prod_{i=1}^{l} SA_{i,j}^{c_i}) \bmod N \\ &= (\prod_{i=1}^{l}(SB_{i,j}/SA_{i,j})^{c_i} \bmod N. \end{aligned}$$

Observe that the ratio of re-signature to signature does not yield the resigning key. Using the re-signature key the proxy can turn Alice's signatures into Bob's and Bob's to Alice's by just inverting the ratio of signatures. Thus the scheme here is bidirectional.
The signature generated by Signature generation algorithm is provided as one of the inputs to the Re-Sign algorithm. When we observe the equations of Signature generation and Re-Sign algorithms, we can say that both are generating signatures of the form $\langle j,(Y,Z)\rangle$ (Bellare-Miner signatures). Thus, signatures generated by either the Sign or ReSign algorithms can be taken as input to ReSign. This property when applied repeatedly can be used to translate Bob's signature to Carol's signature using the Re-sign key $rk_{B\to C,j}$, Carol's signature to Dick's signature using the Re-sign key $rk_{C\to D,j}$ and so on. Therefore we claim that a message can

be re-signed several times which is the property of multi-use scheme. This Bidirectional Multi-use scheme is a Transitive scheme as shown below:
To translate Alice's signature to Bob's signature we use,

$$rk_{A\rightarrow B,j} = (rk_{1,j}^{AB}, \ldots, rk_{l,j}^{AB})$$

where $rk_{i,j}^{AB} = SB_{i,j}/SA_{i,j} \bmod N$.
To translate Bob's signature to Carols's signature we use,

$$rk_{B\rightarrow C,j} = (rk_{1,j}^{BC}, \ldots, rk_{l,j}^{BC})$$

where $rk_{i,j}^{BC} = SC_{i,j}/SB_{i,j} \bmod N$.
To translate Alice's signature to Carols's signature we are required to have,

$$rk_{A\rightarrow C,j} = (rk_{1,j}^{AC}, \ldots, rk_{l,j}^{AC})$$

where $rk_{i,j}^{AC} = SC_{i,j}/SA_{i,j} \bmod N$. Note that $rk_{i,j}^{AC} = SC_{i,j}/SA_{i,j} = (SC_{i,j}/SB_{i,j}).(SB_{i,j}/SA_{i,j}) = rk_{1,j}^{BC}.rk_{1,j}^{AC} \bmod N$

6) **Signature Verification:** A claimed signature $\langle j, (Y, Z)\rangle$ for the message M in time period j is accepted if

$$Z^{2^{(T+1-j)}} = Y \prod_{i=1}^{l} UA_i^{c_i} \bmod N \qquad (3)$$

where $c_1, \ldots, c_l = H(j, Y, M)$, and rejected otherwise. Notice that since

$$\begin{aligned} Z^{2^{(T+1-j)}} &= (R(\prod_{i=1}^{l} SA_{i,j}^{c_i}))^{2^{(T+1-j)}} \bmod N \\ &= Y.(\prod_{i=1}^{l} SA_{i,0}^{2^{(T+1)}c_i}) \bmod N \\ &= Y.\prod_{i=1}^{l} UA_i^{c_i} \bmod N. \end{aligned}$$

a signature by an honest signer with the secret key will be accepted.

C. Multi-use Unidirectional Proxy Re-signature Scheme

To address the first open problem of Ateniese and Hohenberger, we propose a new construction for multi-use (i.e. the translation of signatures can be performed in sequence and multiple times by distinct proxies) unidirectional (i.e. the proxy information allows translating signatures in only one direction) proxy re-signature scheme using the property of forward-security. Our [22] forward-secure proxy re-signature scheme, based on the hardness of factoring, translates one persons signature to another persons signature and additionally facilitates the signers as well as the proxy to guarantee the security of messages signed in the past even if their secret key is exposed today (property of forward-security). The scheme also satisfies the properties viz. private proxy, transparent, unlinkable, key optimal, interactive, non-transitive and temporary. With a minor change in resigning key, we can make the scheme to behave as a multi-use bidirectional scheme.

The key generation, key evolution and signature generation algorithms are same as the ones used in Forward-Secure Multi-use Uni-directional Proxy Re-Signature Scheme discussed in Section 6.4.1. The other algorithms are given below:

1) **Re-Signature Key Generation (ReKey):** On input two secret keys $SKA_j = (SA_{1,j}, \ldots, SA_{l,j}, N_A, j)$ and $SKB_{j+1} = (SB_{1,j+1}, \ldots, SB_{l,j+1}, N_B, j+1)$, the re-signature key, $rk_{A\rightarrow B,j} = (rk_{1,j}, \ldots, rk_{l,j})$ is computed as $rk_{i,j} = SB_{i,j+1}/SA_{i,j} \bmod N$ where $i = 1, \ldots, l; j = 1, \ldots, T-1$.
Observe that the key $rk_{A\rightarrow B}$ can be securely generated as follows:
   a) The proxy sends a random $r \in Z_N$ to Alice.
   b) Alice sends $(r/SA_{1,j}, \ldots, r/SA_{l,j})$ to Bob.
   c) Bob sends $(r(SB_{1,j+1}/SA_{1,j}), \ldots, r(SB_{l,j+1}/SA_{l,j})$ to the proxy.
   d) The proxy recovers $(SB_{1,j+1}/SA_{1,j}, \ldots, SB_{l,j+1}/SA_{l,j})$.
2) **Re-Sign (ReSign):** On input a re-signature key $rk_{A\rightarrow B,j}$, a public key PKA, a signature $\langle j, (Y, Z)\rangle$, and a message M, we check if Verify$(PKA, m, \langle j, (Y, Z)\rangle)$ = 1. If so, we set

$$Z' = Z \prod_{i=1}^{l} rk_{i,j}^{c_i} \bmod N,$$

where $c_1, \ldots, c_l = H(j, Y, M)$ and output the signature $\langle j+1, (Y, Z')\rangle$, otherwise we output an error message. Observe that,

$$\begin{aligned} Z' &= Z.\prod_{i=1}^{l} rk_{i,j}^{c_i} \bmod N \\ &= R\prod_{i=1}^{l} SA_{i,j}^{c_i}.(\prod_{i=1}^{l} SB_{i,j+1})^{c_i}/(\prod_{i=1}^{l} SA_{i,j})^{c_i} \\ &\quad \bmod N \\ &= R\prod_{i=1}^{l} SB_{i,j+1}^{c_i}, \end{aligned}$$

which shows that the signature $\langle j+1, (Y', Z')\rangle$ is Bob's signature. Thus, Re-Sign has translated Alice's signature into Bob's signature.
Even here, just as in BBS scheme, our scheme computes the resigning key as the ratio of secret keys of Alice and Bob, but the resigning key cannot be computed using the signature/re-signature pair as shown below:
Let the re-signature key be $rk_{A\rightarrow B,j} = (rk_{1,j}, \ldots, rk_{l,j})$ where $rk_{i,j} = SB_{i,j+1}/SA_{i,j} \bmod N$.
Let $\langle j, (Y, Z)\rangle$ and $\langle j, (Y, Z')\rangle$ be the signature/re-signature pair where, $Z = R\prod_{i=1}^{l} SA_{i,j}^{c_i} \bmod N_A$

and $Z' = R\prod_{i=1}^{l} SB_{i,j+1}^{c_i} \bmod N$. Then

$$\begin{aligned} Z'/Z &= (R\prod_{i=1}^{l} SB_{i,j+1}^{c_i} \bmod N)/(R\prod_{i=1}^{l} SA_{i,j}^{c_i} \bmod N) \\ &= (\prod_{i=1}^{l} SB_{i,j+1}^{c_i}/\prod_{i=1}^{l} SA_{i,j}^{c_i}) \bmod N \\ &= (\prod_{i=1}^{l} (SB_{i,j+1}/SA_{i,j})^{c_i} \bmod N. \end{aligned}$$

Observe that the ratio of re-signature to signature does not yield the resigning key.

In the protocol indicated in the Re-Signature Key Generation, Alice uses her secret key of j^{th} time period while Bob uses his secret key of $(j+1)^{th}$ time period in the computation. Thus Alice's signature in the j^{th} time period is converted into Bob's signature in the $(j+1)^{th}$ time period. Also, Bob's signature gets verified in the $(j+1)^{th}$ time period but not in j^{th} time period. By choosing Bob's $(j+1)^{th}$ time period secret key and Alice's j^{th} time period secret key we are able to give the Unidirectional property (re-signature keys can only be used for delegation in one direction) to our scheme. This is explained below.

The re-signature key used to translate Alice's signature to Bob's signature is $rk_{A\to B,j} = (rk_{1,j}, \ldots, rk_{l,j})$ where $rk_{i,j} = SB_{i,j+1}/SA_{i,j} \bmod N$. And, the re-signature key required to translate Bob's signature to Alice's signature is $rk_{B\to A,j} = (rk_{1,j}, \ldots, rk_{l,j})$ where $rk_{i,j} = SA_{i,j+1}/SB_{i,j} \bmod N$. We observe that $rk_{B\to A,j}$ cannot be obtained from $rk_{A\to B,j}$ as the proxy has access to ratio of $SB_{i,j+1}/SA_{i,j}$ but not to individual secret key components $SA_{i,j+1}$ and $SB_{i,j}$.

The signature generated by Signature generation algorithm is provided as one of the inputs to the Re-Sign algorithm. When we observe the equations of Signature generation and Re-Sign algorithms, we can say that both are generating signatures of the form $\langle j, (Y, Z)\rangle$ (Bellare-Miner signatures). Thus, signatures generated by either the Sign or ReSign algorithms can be taken as input to ReSign. This property when applied repeatedly can be used to translate Bob's signature to Carol's signature using the Re-sign key $rk_{B\to C,j}$, Carol's signature to Dick's signature using the Re-sign key $rk_{C\to D,j}$ and so on. Therefore we claim that a message can be re-signed several times which is the property of multi-use scheme.

This Unidirectional Multi-use scheme is also Non-Transitive.

To translate Alice's signature to Bob's signature we use, $rk_{A\to B,j} = (rk_{1,j}^{AB}, \ldots, rk_{l,j}^{AB})$
where $rk_{i,j}^{AB} = SB_{i,j+1}/SA_{i,j} \bmod N$.

To translate Bob's signature to Carols's signature we use, $rk_{B\to C,j} = (rk_{1,j}^{BC}, \ldots, rk_{l,j}^{BC})$
where $rk_{i,j}^{BC} = SC_{i,j+1}/SB_{i,j} \bmod N$.

To translate Alice's signature to Carols's signature we are required to have, $rk_{A\to C,j} = (rk_{1,j}^{AC}, \ldots, rk_{l,j}^{AC})$
where $rk_{i,j}^{AC} = SC_{i,j+1}/SA_{i,j} \bmod N$. From the above, $rk_{i,j}^{AC}$ cannot be obtained from $rk_{i,j}^{AB}$ and $rk_{i,j}^{BC}$.

3) **Signature Verification:** As for verification, a claimed signature $\langle j, (Y, Z)\rangle$ for the message M in time period j is accepted if

$$Z^{2^{(T+1-j)}} = Y\prod_{i=1}^{l} UA_i^{c_i} \bmod N \qquad (4)$$

where $c_1, \ldots, c_l = H(j, Y, M)$, and rejected otherwise.

D. Applications in e-banking

As most banking applications require the consent of the signer, we have opted for an interactive method of computing the re-signing key that is, proxy, delegator and delegatee are involved in the computation of the re-signing key.

1) Loan Sanctioning process: In this process a number of bank officials are involved at various stages, right from verifying the records to sanctioning the loan. At every stage the concerned official is required to sign the loan application indicating that the documents given in support of the loan is in accordance with the bank guidelines. As each official signs independently, there is possibility that the officials sign for different data. Also, all the signatures are verified at the end before sanctioning the loan. The problem here is we need to maintain signatures and the public keys of all the officials until the sanction of the loan.

We address this problem using proxy re-signatures. Let us assume that there are four officials, A, B, C and D. The resigning keys, $rk_{A\to B}$, $rk_{B\to C}$ and $rk_{C\to D}$ between the officials are computed. Initially Official A verifies the loan documents pertaining to his section, signs the loan application as s_A and passes the loan application and the documents to Official B. The signature s_A is verified by Official B. He next verifies the loan documents pertaining to his section and applies the Re-Sign algorithm which converts official A's signature to his own signature, s_B. By doing this, along with Official A Official B has become responsible for verifying the documents as the signature of Official B is not created independently but by using the signature of Official A. In this way, at every stage of loan processing, one official's signature is replaced by another official's signature. In the end, only Official D's signature will be on the loan application where Official D can be assumed as the Manager of

the bank. By using re-signatures following are the advantages:

- At every stage of loan processing only one verification with one public key is sufficient.
- Only one signature needs to be stored at any stage.
- The original message cannot be changed.
- On re-signing, the corresponding official becomes responsible for the completion of the process at that stage.
- At the send, only one signature verification is required instead of four verifications.

2) Frequently changing public keys: A customer of a bank may frequently change his public key due to policy of the organisation or for the sake of security or due to leakage of his secret key. Let (PK_O, SK_O) be the old public key - secret key pair and (PK_N, SK_N) be the new public key - secret key pair of a customer. Sometimes there may be need to verify some old documents which were signed using the old secret key. To handle this situation banks can store the resigning key $rk_{O\to N}$ (this key can be computed whenever the customer opts for change of public key secret key pair) which helps them to translate an old signature signed using SK_O to a new signature which can be verified using the new public key PK_N. This enables to verify old signatures and also all signatures (old or new) using the new public key.
3) Accounts to be operated by a nominee: On many occasions a customer A may be disabled (for a short or long duration) to operate his account. This forces the bank to give power to the nominee B to operate the account. The resigning key $rk_{B\to A}$ is required to be computed by the bank when the account holder declares his nominee. On any transaction done by the nominee B, bank translates the signature to the original account holder's (here A) signature using proxy re-signatures. This translation is not possible without the bank's intervention. By using proxy re-signatures, the bank need not store the public key of the nominee to verify his signature. This facility given to nominee can be revoked at any instant.
4) Transferrable e-cheques: We propose a method for transferring e-cheques with a proxy re-signature[23] having transitive property. Let us assume that there are four persons A, B, C and D and A issues a cheque to B. If B wants to re-issue the same e-cheque to C, B must act as a proxy and compute the re-signing key $rk_{A\to B}$ by communicating with A. Using this key, B can translate A's signature to that of his own. When C re-issues the e-cheque to D, in the same way as B, C act as a proxy and computes the re-signing key $rk_{B\to C}$ and translates B's signatures to that of his own. Before D deposits the e-cheque in his bank, he translates $C's$ to that of his own. The bank verifies the signature of D, which implies the verification of $A's$ signature by virtue of transitivity of proxy signatures. If A also has an account in the same bank, the bank deducts the cheque amount from $A's$ account and credits the same to $D's$ account. If A has an account in a different bank, the bank sends the e-cheque details to that bank, which on verifying A's e-cheque details like account number and cheque number deducts the cheque amount from A's account and sends a message to credit the cheque amount to D's account. Thus, whenever a person wants to re-issue an e-cheque to another person, he can translate the signature of the issuer of the e-cheque existing on the e-cheque to his own signature. Of course, there is additional cost involved in computing the resigning key. But any person who receives the e-cheque needs to verify the signature only with the public key of the person who issued the e-cheque to him.

III. Proxy Re-signature Scheme to translate one type of Signature Scheme to another type

In this section [24] we develop proxy signature scheme for translating the following signatures:

1) Alice's Schnorr Signature to Bob's RSA Signature
2) Alice's ElGamal Signature to Bob's RSA Signature
3) Alice's RSA Signature to Bob's RSA Signature

In these schemes, signatures generated by signature generation algorithm and the re-signature algorithms are indistinguishable. Further, Proxy signer revocation is also possible.

A. Alice's Schnorr Signature Scheme to Bob's RSA Signature Scheme

Following protocols are used to translate Alice's Schnorr Signature to Bob's RSA Signature:

1) Key generation for Schnorr Signatures
2) Key generation for RSA Signatures
3) Re-Signature key generation
4) Schnorr Signature generation
5) Schnorr Signature verification
6) Re-Sign Algorithm
7) RSA Signature verification.

In the following subsections we discuss the above protocols.

1) **Key Generation for Schnorr Signature Scheme:**
   a) Alice chooses a random large prime p such that $p-1 = kq$ for some integer k and large prime q.
   b) She chooses randomly a secret key x in the range $0 \le x \le q-1$ and generator $\alpha \in Z_p^*$ of order q.
   c) She computes $\beta = \alpha^{-x} \bmod q$. Alice's Public key is (p, q, α, β) and Secret key is x.
2) **Key Generation for RSA Signature Scheme:**

a) Bob generates two large distinct random primes p_1 and q_1, each roughly of the same size.
b) He computes $n = p_1.q_1$ and $\phi(n) = (p-1)(q-1)$.
c) He selects a random integer e, $1 < e < \phi(n)$, such that $gcd(e, \phi(n)) = 1$.
d) He computes the unique integer d, $1 < d < \phi(n)$, such that $ed \equiv 1(\mathrm{mod}\phi(n))$.
e) Bob's Public key is (n, e) and Private key is d.

3) **Re-Signature Key Generation (Re-Key):** Four re-signature keys $rk1_{A\to B}$, $rk2_{A\to B}$, $rk3_{A\to B}$ and $rk4_{A\to B}$ are required for this conversion.
a) *The re-signature key $rk1_{A\to B}$ is generated as follows:*
i) Bob randomly chooses $k \in Z$ and sends $k.\phi(n)$ to Alice.
ii) Alice computes $x \equiv x_1 \bmod (k.\phi(n))$ and sends it to proxy. Note that if $x \equiv x_2 \bmod \phi(n)$ then $x_1 \equiv x_2 \bmod \phi(n)$.
iii) Proxy sets re-sign key as $rk1_{A\to B} = x_1 \bmod (k.\phi(n))$.
b) *The re-signature key $rk2_{A\to B}$ is generated as follows:*
i) Bob randomly chooses $l \in Z$ and sends $l.\phi(n)$ to Alice and to Proxy.
ii) Proxy randomly chooses $r \in Z_{l.\phi(n)}$ and sends it to Alice.
iii) Alice computes $r + x \bmod (l.\phi(n))$ and sends it to Bob.
iv) Bob computes $r+x+d \bmod (l.\phi(n))$ and sends it to proxy.
v) Proxy sets the re-sign key as $rk2_{A\to B} = x + d \bmod (l.\phi(n))$.
c) *The re-signature key $rk3_{A\to B}$ is generated as follows:*
i) Proxy sends a random $r \in Z_q$ to Alice.
ii) Alice sends $r + x \bmod (q-1)$ to Bob,
iii) Bob sends $r + x + d \bmod (q-1)$ to the proxy.
iv) Proxy recovers $d+x \bmod (q-1)$ and sets the re-sign key $rk3_{A\to B} = d+x \bmod (q-1)$.
d) *The re-signature key $rk4_{A\to B}$ is generated as follows:*
i) Bob sends $\alpha^d \bmod q$ to Proxy.
ii) Proxy sets the re-sign key $rk4_{A\to B} = \alpha^d \bmod q$.

4) **Schnorr Signature generation:**
a) Alice selects a random integer k, such that $0 \le k \le q-1$.
b) She computes $r = \alpha^k \bmod q$, $v = H(m||r) \bmod q$ and $s = (k + x.v) \bmod q$, where H is a collision-resistant hash function.
c) The signature of Alice on the message m is (v, s), which is sent to Proxy.

5) **Schnorr Signature verification:** On receiving Alice's signature on m, proxy does the following to verify the signature:
a) He computes $v' = H(m||r_v)$ where, $r_v = \alpha^s.\beta^{-v} \bmod q$.
b) He accepts the signature if and only if $v' = v$.

6) **Re-signature (Re-Sign)** On input re-signature keys $rk1_{A\to B}$, $rk2_{A\to B}$, $rk3_{A\to B}$ and Alice's Schnorr signature (v, s) on m, Re-Sign converts Alice's Schnorr signature to Bob's RSA signature in two steps.
- **Step 1:** Compute $\sigma_1 = \alpha^{s-rk3_{A\to B}.v}.(rk4_{A\to B})^v$ and check that $\sigma_1 = 1$.
  Note that this step ensures that Alice uses the same key used during re-signature key generation to sign the message m.
- **Step 2:** If $\sigma_1 = 1 \bmod q$, then compute

$$\sigma_2 = \mathcal{R}(m)^{rk2_{A\to B}}.\mathcal{R}(m)^{-rk1_{A\to B}} \bmod n,$$

  where $\mathcal{R}$ is the public redundancy function [25].
  Note that $\sigma_2 = \mathcal{R}(m)^d \bmod n$ is Bob's RSA signature.

  Thus, Re-Sign has translated Alice's Schnorr signature to Bob's RSA signature.

7) **RSA Signature verification:** Any verifier can verify the RSA signature generated by Re-sign as follows:
a) Compute $\mathcal{R}(m) = C^e \bmod n$.
b) Verify $m = \mathcal{R}^{-1}(\mathcal{R}(m)) \bmod n$.

B. Alice's ElGamal Signature to Bob's RSA Signature Scheme

Following protocols are used to translate Alice's ElGamal Signature to Bob's RSA Signature:

1) Key generation for ElGamal Signatures
2) Key generation for RSA Signatures
3) Re-Signature key generation
4) ElGamal Signature generation
5) ElGamal Signature verification
6) Re-Sign Algorithm
7) RSA Signature verification.

In the following subsections we discuss the above protocols.

1) **Key generation for ElGamal Signatures:**
a) Alice generates a large random prime p and a generator α of the multiplicative group Z_p^*.
b) She select a random integer s, $1 \le s \le p-2$. s is the secret key.
c) She computes the public key $\beta = \alpha^s \bmod p$.

2) **Key generation for RSA Signatures:**
a) Bob generates two large distinct random primes p_1 and q_1, each roughly the same size.

b) He computes $n = p_1.q_1$ and $\phi(n) = (p-1)(q-1)$.
c) He select a random integer e, $1 < e < \phi(n)$, such that $gcd(e, \phi(n)) = 1$.
d) He computes the unique integer d, $1 < d < \phi(n)$, such that $ed \equiv 1(\mathrm{mod}\phi(n))$.
e) Bob's Public key is (n, e) and Private key is d.

3) **Re-Signature Key Generation (Re-Key):** Four re-signature keys $rk1_{A\to B}$, $rk2_{A\to B}$, $rk3_{A\to B}$ and $rk4_{A\to B}$ are required for this conversion.
a) *The re-signature key $rk1_{A\to B}$, is generated as follows:*
i) Bob randomly chooses $k \in Z$ and sends $k.\phi(n)$ to Alice.
ii) Alice computes $s \equiv s_1 \bmod (k.\phi(n))$. Note that if $s \equiv s_2 \bmod \phi(n)$ then $s_1 \equiv s_2 \bmod \phi(n)$
iii) Proxy sets resign key as $rk1_{A\to B} = s_1 \bmod (k.\phi(n))$.
b) *The re-signature key $rk2_{A\to B}$ is generated as follows:*
i) Bob randomly chooses $l \in Z$ and sends $l.\phi(n)$ to Alice and to Proxy.
ii) Proxy randomly chooses $r \in Z_{l.\phi(n)}$ and sends it to Alice.
iii) Alice computes $r + s \bmod (l.\phi(n))$ and sends it to Bob.
iv) Bob computes $r + s + d \bmod (l.\phi(n))$ and sends it to proxy.
v) Proxy sets the re-sign key as $rk2_{A\to B} = s + d \bmod (l.\phi(n))$.
c) *The re-signature key $rk3_{A\to B}$ is generated as follows:*
i) Proxy sends a random $r \in Z_p$ to Alice.
ii) Alice sends $r + s \bmod (p-1)$ to Bob.
iii) Bob sends $r + s + d \bmod (p-1)$ to the proxy.
iv) Proxy recovers $s + d \bmod (p-1)$ and sets $rk3_{A\to B} = s + d \bmod (p-1)$.
d) *The re-signature key $rk4_{A\to B}$ is generated as follows:*
i) Bob sends $\alpha^d \bmod p$ to Proxy.
ii) Proxy sets the re-sign key $rk4_{A\to B} = \alpha^d \bmod p$.

4) **ElGamal Signature generation:**
a) Alice selects a random secret integer k, $1 \le k \le p-2$ with $gcd(k, p-1) = 1$.
b) She computes $y_1 = \alpha^k \mod p$.
c) She computes $y_2 = (H(m) - s.y_1)k^{-1} \mod (p-1)$, where H is a collision-resistant hash function.
d) The signature (y_1, y_2) on message m is sent to the Proxy.

5) **ElGamal Signature verification:** The proxy verifies the received ElGamal signature as follows:
a) He accepts the signature if $\alpha^{H(m)} = \beta^{y_1} y_1^{y_2} \bmod p$.

6) **Re-signature (ReSign):** On input of re-signature keys $rk1_{A\to B}$, $rk2_{A\to B}$ and $rk3_{A\to B}$ and Alice's ElGamal signature (y_1, y_2) on m, Re-Sign algorithm converts Alice's ElGamal signature to Bob's RSA signature in two steps:
- **Step 1**: Compute $\sigma_1 = y_1^{y_2}.\alpha^{-H(m)}.\alpha^{(d+s).y_1}.\alpha^{-d.y_1} \bmod p$ and check that $\sigma_1 = 1$. Note that this step ensures that Alice uses the same key used during re-signature key generation to sign the message m.
- **Step 2** If $\sigma_1 = 1 \bmod p$, then compute
$$\sigma_2 = \mathcal{R}(m)^{rk2_{A\to B}}.\mathcal{R}(m)^{-rk1_{A\to B}} \bmod n,$$
where $\mathcal{R}$ is the public redundancy function [25].
Note that $\sigma_2 = \mathcal{R}(m)^d \bmod n$
Thus, Re-Sign has translated Alice's ElGamal signature into Bob's RSA signature.

7) **RSA Signature verification:** Any verifier can verify the RSA signature generated by Re-sign as follows:
a) Compute $\mathcal{R}(m) = C^e \bmod n$.
b) Verify $m = \mathcal{R}^{-1}(\mathcal{R}(m)) \bmod n$.

C. Alice's RSA Signature Scheme to Bob's RSA Signature Scheme

In literature, to the best of our knowledge, there are no proxy re-signature schemes for primitive signature schemes like RSA, ElGamal, Schnorr, etc., though such schemes are commercially used. Therefore, we present construction of a scheme which converts Alice's RSA signature to Bob's RSA signature. We construct this by generating suitable proxy re-sign keys which are computed by establishing communication among delegatee, proxy signer and the delegator. At no point of conversion the security of RSA signature scheme is compromised.

Following protocols are used to translate Alice's RSA Signature [25] to Bob's RSA Signature:

1) Key generation for Alice and Bob as in RSA Signature Scheme.
2) Re-Signature key generation
3) Alice's RSA Signature generation
4) Alice's RSA Signature verification
5) Re-Sign Algorithm
6) Bob's RSA Signature verification

We discuss these protocols in the following subsections.

1) **RSA Key generation:** Let (n_a, e_a) be the Public key and (n_a, d_a) be the Private key of Alice. Let (n_b, e_b) be the Public key and (n_b, d_b) be the Private key of Bob. We assume that n_a is less than n_b. For Alice (Bob), let $\mathcal{M}_a$ ($\mathcal{M}_b$) be the message space, $\mathcal{M}_{Sa}$ ($\mathcal{M}_{Sb}$) be the signing space and $\mathcal{R}_a$ ($\mathcal{R}_b$) be the public redundancy function which gives a $1-1$ mapping from $\mathcal{M}_a$ ($\mathcal{M}_b$) to $\mathcal{M}_{Sa}$ ($\mathcal{M}_{Sb}$).

2) **Re-Signature Key Generation (ReKey):** The re-signature key $rk_{A\to B}$, is generated as follows:
   a) Bob randomly chooses $l \in Z_{n_b}$ and sends it to Alice secretly.
   b) Bob computes $(d_b - l) \bmod n_b$ and sends it to Proxy.
   c) Proxy sets the re-sign key as $rk_{A\to B} = d_b - l \bmod n_b$.
3) **Alice's RSA Signature generation:**
   a) Alice computes $C_a = \mathcal{R}_a(m)^{d_a} \bmod n_a$. She also computes $\mathcal{R}_b(m)^l \bmod n_b$.
   b) Alice's signature on m is C_a and additionally $R_b(m)^l$ is sent to proxy.
4) **Alice's RSA Signature verification:** The proxy verifies the received RSA signature as follows:
   a) Compute $\mathcal{R}_a(m) = C_a^{e_a} \bmod n_a$.
   b) Verify $m = \mathcal{R}_a^{-1}(\mathcal{R}_a(m)) \bmod n_a$
5) **Re-Sign (ReSign):** On input of re-signature key $rk_{A\to B}$, Alice's RSA signature C_a on m and additionally $\mathcal{R}_b(m)^l$, Re-Sign converts Alice's RSA signature to Bob's RSA signature C_b on m. Taking the value of m from Alice's RSA signature verification,

$$\begin{aligned} C_b &= \mathcal{R}_b(m)^l.\mathcal{R}_b(m)^{rk_{A\to B}} \bmod n_b \\ &= \mathcal{R}_b(m)^l.\mathcal{R}_b(m)^{d_b - l} \bmod n_b \\ &= \mathcal{R}_b(m)^{d_b} \bmod n_b. \end{aligned}$$

   Thus, Re-Sign has translated Alice's RSA signature into Bob's RSA signature.
6) **RSA Signature verification:** Any verifier can verify the RSA signature generated by Re-sign as follows:
   a) Compute $\mathcal{R}_b(m) = (C_b)^{e_b} \bmod n_b$.
   b) Verify $m = \mathcal{R}_b^{-1}(\mathcal{R}_b(m)) \bmod n_b$.

D. Comparison among the various proxy re-signature schemes

Table 1 gives the comparison among the various proxy re-signature schemes. BBS is the scheme of Blaze, Bleumer, and Strauss, AH_b is the bidirectional scheme of Ateniese and Hohenberger, AH_u is the unidirectional scheme of Ateniese and Hohenberger, LV is Libert-Vergnaund scheme, O_U is our Unidirectional scheme, O_B is our Bi-directional scheme and O_R is the RSA Proxy-resignature scheme.

E. Conversion of one type of Signature to another type : Applications

1) If we have schemes which convert different types of signatures to one type of signature, for example all types of signatures are converted to RSA signatures, then signatures can be easily aggregated and a single verification done to verify all the signatures which are originally of different type signatures.

No.	Property	BBS	AH_b	AH_u	LV	O_U	O_B	O_R
1	Unidirect	No	No	Yes	Yes	Yes	No	Yes
2	Multi-use	Yes	Yes	No	Yes	Yes	Yes	Yes
3	Prv. proxy	No	Yes	No	No	Yes	Yes	Yes
4	Transparent	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Key Optimal	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	Non-interact	No	No	Yes	Yes	No	No	No
7	Non-transit	No	No	Yes	Yes	Yes	No	Yes
8	Temporary	No	No	Yes	No	Yes	Yes	Yes
9	Forward-sec	No	No	No	No	Yes	Yes	No

TABLE I.
COMPARISON AMONG THE VARIOUS PROXY RE-SIGNATURE SCHEMES

2) By providing a variety of conversion schemes, a user can sign using any signature scheme and the required party can convert it into the signature type that it requires.
3) Using signature conversion schemes, documents can be easily transferred across organisations following different signature schemes.
4) Suppose Alice's ElGamal Signature is converted to Bob's RSA signature, Bob's signature is publicly available and there are chances of Bob's secret key being exposed than Alice's secret key. Thus the original signer Alice's secret key is more secure than Bob's secret key.

IV. ANALYSIS AND SECURITY FOR THE PROPOSED SCHEMES

Here we analyse the properties satisfied by the proposed Proxy Re-signature schemes.

A. Properties of the New Proxy Re-signature schemes

1) Unidirectional: The re-signature keys allows the proxy to turn Alice's signatures into Bob's, but not Bob's into Alice's. This property allows for applications where the trust relationship between two parties is not necessarily mutual. Schemes that do not have this property are called bidirectional.
2) A message can be re-signed a polynomial number of times. That is, signatures generated by either the Sign or ReSign algorithms can be taken as input to ReSign.
3) Private Proxy: In a private proxy scheme, the re-signature keys are kept secret by an honest proxy. Thus, a single proxy may control which signatures get translated.
4) Transparent: The proxy is transparent in the scheme, meaning that a user may not even know that a proxy exists. More formally, we mean that the signatures generated by Bob on a message m using the Sign algorithm are computationally indistinguishable from the signatures on m generated by the proxy as the output of ReSign. Notice that this implies that the input and the corresponding output of the ReSign algorithm cannot be linked to each other.
5) Key Optimal: Alice is only required to protect and store a small constant amount of secret data

(i.e., secret keys) regardless of how many signature delegations she gives or accepts. Here, we want to minimize the safe storage cost for each user.

6) Interactive: The proxy creates the re-signature keys by interacting with Bob (the delegator) and Alice (the delegatee).
7) Non-transitive: According to the property of transitivity, from $rk_{A\rightarrow B}$ and $rk_{B\rightarrow C}$, the proxy must be able to produce $rk_{A\rightarrow C}$. The new schemes are non-transitive as we are considering conversion of schemes between different public key cryptosystems and still there are no schemes to convert between RSA and ElGamal/Schnorr.
8) Temporary: Whenever a party delegates some of her rights to another party, there is always the chance that she will either need or want to revoke those rights later on. We have discussed this under proxy signer revocation.

B. Security of our schemes

Our security model protects users from two types of attacks: those launched from parties outside the system (External Security) and those launched from parties inside the system such as the proxy, another delegation partner or collusion between them (Internal Security).

External Security: Our security model protects a user from adversaries outside the system. This is equivalent to adaptive chosen-message attack where an adversary cannot create a new signature even for a previously signed message. For a non-zero $n \in poly(k)$ and algorithm A,

$Pr[(pk_i, sk_i) \leftarrow KeyGen(1^k)_{i\in}, (t, m, \sigma) \leftarrow A^{O_{sign}(.,.),O_{resign}(.,.,.,.)}:$
$Verify(pk_t, m, \sigma) = 1] < 1/poly(k)$

where the oracle O_{sign} takes as input an index j and a message $m \in M$, and produces the output of $Sign(sk_j, m)$; the oracle O_{resign} takes as input two distinct indices i, j, message m, and signature σ and produces the output of $Resign$. Here the proxy ie required to keep the re-signature keys private.

Internal Security: If the delegator and delegatee are both honest, then:

1) the proxy cannot produce signatures for the delegator unless the message was first signed by the delegatee and
2) the proxy cannot create any signature for the delegatee.

V. Proxy Signer Revocation

In all the constructions discussed above, we can revoke the proxy signer from performing conversion of signatures from one type of signature scheme to another type by including the expiry date as part of the signature. If the current date is greater than the expiry date, the verifier stops the verification process. If a proxy signer is to be revoked before the expiry date, a proxy revocation list can be maintained in which the public key of proxy signers to be revoked are entered. The entry is retained only till the expiry date and later removed. This helps to maintain a small list of revoked proxy signers. The verifier performs the verification of the signature received from the proxy signer only if his public key is not available in the proxy revocation list and the current date is less than or equal to the expiry date.

VI. Conclusion

We have proposed two schemes for the open challenges in proxy re-signatures. The first proposed scheme is the design of multi-use unidirectional proxy re-signature scheme in which one person's signature is translated to another person's signature and additionally facilitating the signers as well as the proxy to guarantee the security of messages signed in the past even if their secret key is exposed today. Our scheme is a multi-use unidirectional scheme where the proxy is able to translate in only one direction and signatures can be re-translated several times. The second scheme convert Schnorr/ElGamal/RSA to RSA signatures. The signatures generated by regular signature generation algorithm and the proposed re-signature schemes are indistinguishable.

References

[1] Blaze and Bleumer and Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology EUROCRYPT, volume 1403 of LNCS, Springer-Verlag, 241-256*, Mar. 1998.
[2] G. Ateniese, S. Hohenberger, "Proxy re-signatures: new definitions, algorithms, and applications," in *ACM CCS, pages 310319, ACM Press*, Mar. 2005.
[3] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signature: Delegation of the power to sign messages," *IEICE Trans. Fundamentals, pp. 13381353*, no. 9, Sept. 1996.
[4] M. Mambo, K. Usuda, E. Okamoto, "Proxy signatures for delegating signing operation," in *3rd ACM Conference on Computer and Communications Security, pp. 48-57. ACM Press*, Mar. 1996.
[5] A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," in *http://eprint.iacr.org/2003/096*, Oct. 2003.
[6] B. Lee, H. Kim and K. Kim., "Strong proxy signature and its applications," in *the 2001 Symposium on Cryptography and Information Security, Vol. 2/2, pp. 603-608. Oiso, Japan, 23-26,pages 127144,*, Jan. 2001.
[7] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," in *Information and Communications Security, LNCS 1334, pp. 223-232. Springer-Verlag*, Mar. 1997.
[8] Benoit Libert and Damien Vergnaud, "Multi-Use Unidirectional Proxy Re-Signatures," in *In CCS Proceedings of the 15th ACM conference on Computer and communications security,Alexandria, VA, USA*, Oct. 2008.
[9] D. Boneh, B. Lynn, H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology-Crypto proceedings,volume 2248 of LNCS, pp. 514-532, Springer-Verlag*, Mar. 2002.
[10] M. Bellare, P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS, pages 6273, ACM Press*, Mar. 1993.
[11] Xiaoming Hu; Zhe Zhang; Yinchun Yang, "Identity Based Proxy Re-Signature Schemes without Random Oracle," in *Computational Intelligence and Security, pp. 256-259*, Mar. 2009.

[12] Sherman S.M. Chow, Raphael C W Phan, "Proxy Re-signatures in the Standard Model," in *ISC, LNCS 5222, pp. 260-276, Springer-Verlag Berlin heldelberg*, Mar. 2008.
[13] Yu Qiao Deng, "A Blind Proxy Re-Signatures Scheme Based on Random Oracle," in *Advanced Research on Industry, Information System and Material Engineering*, Feb. 2011.
[14] Xiaodong Yang, Caifen Wang, Yulei Zhang, Weiyi Wei, "A New Forward-secure Threshold Proxy re-signature scheme," in *IEEE International Conference on Network Infrastructure and Digital Content, pp. 566 - 569*, Mar. 2009.
[15] Jun Shao, Zhenfu Cao, Licheng Wang and Xiaohui Liang, "Proxy Re-signature Schemes without Random Oracles," in *Indocrypt, LNCS 4859, pp. 197-209, Springer Verlag*, Mar. 2007.
[16] Anderson, R., "Forward Security," in *Fourth Annual Conference on Computer and Communications Security*, Mar. 1997.
[17] Bellare, M., Miner, S., "A Forward-Secure Digital Signature Scheme," in *Advances in Cryptology-Crypto proceedings, Lecture notes in Computer Science, Vol. 1666. Springer-Verlag*, Mar. 1999.
[18] Abdalla,M., Reyzin,L, "A New Forward-Secure Digital Signature Scheme," in *ASIACRYPT, LNCS 1976, pp. 116-129. Springer-Verlag,116-129*, Mar. 2000.
[19] Krawczyk, H., "Simple forward-secure signatures from any signature scheme," in *the 7th ACM Conference on Computer and Communications Security, ACM, 108-115*, Mar. 2000.
[20] Itkis, G., Reyzin, L., "Forward-secure signatures with optimal signing and verifying," in *CRYPTO, LNCS 2139, Springer-Verlag, 332-354*, Mar. 2001.
[21] Kozlov, A, Reyzin, L., "Forward-Secure Signatures with Fast Key Update," in *Security in Communication Networks, LNCS 2576, Springer-Verlag, (241-256)*, Mar. 2002.
[22] N.R. Sunitha and B.B.Amberker, "Multi-use Unidirectional Forward-Secure Proxy Re-Signature scheme," in *IEEE Workshop on Collaborative Security Technologies, Bangalore, India*, Dec. 2009.
[23] N.R.Sunitha B.B.Amberker and Prashant Koulgi, "Transferable e-cheques using Forward-Secure Multi-signature Scheme," in *The World Congress on Engineering and Computer Science, 24-26, San Francisco, USA*, Oct. 2007.
[24] N.R. Sunitha and B.B. Amberker, "Proxy re-signature scheme that translates one type of signature scheme to another type of signature scheme," in *Third International Conference on Network Security and Applications, Chennai, India, Communications in Computer and Information Science Series , Springer Verlag*, July 2010.
[25] A.Menezes, P.Van Orschot and S.Vanstone, "Handbook of applied cryptography," in *CRC Press, 1996*.

Biographies

N.R.Sunitha obtained her Ph.D from Visvesvaraya Technological University, Belgaum, Karnataka, India. She is presently working as Associate Professor in the Department of Computer Science & Engineering, Siddaganga Institute of Technology, Tumkur, India. Her area of Research is Information Security.

B.B.Amberker obtained his Ph.D from the Department of Computer Science & Automation, IISc., Bangalore, India. He is presently working as Professor in the Department of Computer Science & Engineering, National Institute of Technology, Warangal, India. His area of Research is Cryptography.