A Signcryption Scheme Based on Integer Factorization

Ron Steinfeld and Yuliang Zheng

Laboratory for Information and Network Security,
School of Network Computing,
Monash University,
Frankston 3199, Australia
{ron.steinfeld,yuliang.zheng}@infotech.monash.edu.au

Abstract. Signcryption is a public-key cryptographic primitive introduced by Zheng, which achieves both message confidentiality and non-repudiatable origin authenticity, at a lower computational and communication overhead cost than the conventional 'sign-then-encrypt' approach. We propose a new signcryption scheme which gives a partial solution to an open problem posed by Zheng, namely to find a signcryption scheme based on the integer factorization problem. In particular, we prove that our scheme is existentially unforgeable, in the random oracle model, subject to the assumption that factoring an RSA modulus $N = pq$ (with p and q prime) is hard even when given the additional pair (g, S), where $g \in \mathbb{Z}_N^*$ is an asymmetric basis of large order less than a bound $S/2 \ll \sqrt{N}$.

1 Introduction

Confidentiality and non-repudiatable origin authenticity of transmitted information are important requirements in many applications of cryptography. The conventional approach to meeting these goals is the 'sign-then-encrypt' technique, where the message originator produces a digital signature on the message using his secret key and then encrypts the signed message with the recipient's public key.

Several years ago, Zheng [17] introduced a public-key cryptographic primitive called signcryption, which achieves both confidentiality and non-repudiatable origin authenticity at a lower computational and communication overhead cost than the 'sign-then-encrypt' technique. The original signcryption schemes were based on the discrete logarithm problem $DLP(p, q)$ in a multiplicative subgroup of prime order q in $\mathbb{Z}_p^*$, for p prime. These signcryption schemes are currently regarded as secure because there is no known efficient algorithm for solving $DLP(p, q)$. However, the fact that $DLP(p, q)$ is currently not a *provably* difficult problem (and hence vulnerable to algorithmic breakthroughs) motivates the search for signcryption schemes based on other difficult problems which are computationally independent of $DLP(p, q)$. Consistent with this approach,

Zheng posed, in his original paper [17], the open problem of finding a signcryption scheme based on the integer factorization problem, which appears to be computationally independent of $DLP(p, q)$.

In this paper, we propose a partial solution to Zheng's open problem. In particular, we modify an efficient signature scheme recently proposed [11] by Pointcheval to obtain a new signcryption scheme heuristically based on the *Composite Discrete Logarithm* problem $CDLP(N, g, S, y)$ with suitably chosen parameters, namely: Given (N, g, S, y), where N is a composite RSA modulus $N = pq$ (with p and q large primes, $p-1$ and $q-1$ non-smooth), g is an element of $\mathbb{Z}_N^*$ satisfying $1 \ll \mathrm{Ord}_{\mathbb{Z}_N^*}(g) < S/2 \ll \sqrt{N}$ (with $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$ having no small prime factors besides 2) and $m_2(\mathrm{Ord}_{\mathbb{Z}_p^*}(g)) \neq m_2(\mathrm{Ord}_{\mathbb{Z}_q^*}(g))$ (where $m_2(z)$ denotes the largest α such that 2^α divides z), $y \in < g >$ (where $< g >$ denotes the multiplicative subgroup of $\mathbb{Z}_N^*$ generated by g), find $x \in \mathbb{N}$ such that $g^x = y \bmod N$.

For the above choice of parameters, $CDLP(N, g, S, y)$ is harder than the problem of factoring N given (N, g, S) because (by Lemma 6 in Appendix) a non-zero multiple of $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$ reveals the factorization of N, and it is easy to compute such a multiple using an oracle which solves $CDLP(N, g, S, y)$. We conjecture that, assuming the symmetric cipher and one-way hash functions used by our scheme have no individual weaknesses, any security break of our scheme is harder than factoring N. To support this claim, we prove, in the random oracle model, that under the assumption that factoring N given (N, g, S) is intractable, our scheme is existentially unforgeable under an adaptively chosen message attack. The confidentiality of our scheme is based on the Diffie-Hellman problem in $\mathbb{Z}_N^*$ with base g, which is also believed to be harder than factoring N.

We call our proposal a 'partial' solution to Zheng's problem for the following reasons:

(1) Our scheme's unforgeability is proven with respect to a non-standard factorization problem, due to the additional knowledge of the element $g \in \mathbb{Z}_N^*$ whose order is large but smaller than a known bound $S \ll \sqrt{N}$. However, with a suitable choice of parameters to make direct $CDLP$ attacks infeasible, the problem appears to be as hard as the standard RSA modulus factorization problem.

(2) Our scheme's security is proven with respect to the problem of factoring a *common* public modulus N shared by all users. On the other hand, generating the common public parameters of our scheme requires knowledge of (p, q), the factors of N. Therefore the authority generating the common parameters must be trusted by all users to not be a potential attacker and to not reveal (p, q) to any potential attacker. We remark that this constraint is much weaker than that imposed by 'identity' schemes based on factorization (eg Fiat-Shamir [6]) since in our scheme (p, q) need not be kept by the trusted authority for generating secret keys for new 'identities'.

The rest of the paper is organized as follows. In Sect. 2 we review related past work. In Sect. 3, we detail our new signcryption scheme $\mathcal{SCF}$. In Sect. 4, we suggest practical values our scheme's parameters and compare its efficiency with

earlier schemes. In Sect. 5, we present definitions of security notions for general signcryption schemes, and state these properties for our particular scheme. Due to lack of space some proofs are omitted and included in the full paper, available from the authors. Finally, Sect. 6 contains concluding remarks.

2 Background

In 1989, Schnorr [15] proposed well-known efficient 3-move zero-knowledge identification and a corresponding signature scheme based on the difficulty of computing discrete logarithms in a subgroup of $\mathbb{Z}_p^*$ (for p a public prime) generated by public element $g \in \mathbb{Z}_p^*$ of public prime order q. We quickly review the Schnorr signature here to allow the following schemes to be explained. The signer Alice picks a random secret key $s_A \in \mathbb{Z}_q^*$ and publishes her public key $v_A = g^{-s_A} \bmod p$. To sign a message m, Alice picks a random $r \in \mathbb{Z}_q^*$ and computes the signature consisting of the pair (e, y), with $e = \mathsf{H}(m, g^r \bmod p)$, and $y = r + e \cdot s_A \bmod q$, where $\mathsf{H}(.)$ denotes a 'one-way' hash function. Bob verifies that the pair (e', y') is Alice's signature on message m' by checking that $e' = \mathsf{H}(m', g^{y'} v_A^{e'})$. An attractive feature of Schnorr's scheme is its low *on-line* computational cost, since the time-consuming modular exponentiation $g^r \bmod p$ is message-independent and can be performed off-line.

In 1991, Girault [8] proposed a variant of Schnorr's scheme, replacing the group $\mathbb{Z}_p^*$ by the group $\mathbb{Z}_N^*$, where $N = pq$ (p,q prime). In addition, Girault proposed the use of a subgroup generator $g \in \mathbb{Z}_N^*$ of maximal order $\lambda(N) = \mathrm{lcm}(p-1, q-1)$. Since Miller's factoring algorithm (see [14]) factors N efficiently when a multiple of $\lambda(N)$ is known, the order of the public generator g must be kept secret from all users. This has the implication for the signing algorithm that the modular reduction in the computation of y cannot be performed. Indeed, Girault proposed to eliminate the modular reduction so y is computed in the integers. This further improves the on-line computational efficiency at the expense of a longer resulting signature. In 1998, Poupard and Stern [13] analysed Girault's scheme and proved its security relative to the discrete logarithm problem in the subgroup $< g >$ of $\mathbb{Z}_N^*$. However (see Sect. 4), their proof does not appear to apply to the efficient variants they propose, in which the size of the secret key space S is much smaller than $\sqrt{N}$. In PKC 2000, Pointcheval [11] proposed a new variant of Girault's scheme in which $S \ll \sqrt{N}$ yielding an efficient scheme as before, but g is chosen differently to allow a proof of security relative to factorization. In particular, Pointcheval proposed to reduce the order of g from $\lambda(N)$ to a large value less than $S/2$, so that at least two secret keys are mapped to each public key. Then the reduction from factorization to breaking the scheme could be performed (in the random oracle model, with Pointcheval and Stern's forking technique [12]) using a variant of Miller's factoring algorithm (which requires an additional condition on the choice of g) and the fact that an attacker cannot distinguish between signatures made, respectively, with two keys mapping to the same public key (i.e. the scheme is *witness indistinguishable*). Our signcryption scheme is based on Pointcheval's approach.

3 New Signcryption Scheme

In this section we present informally our new signcryption scheme $\mathcal{SCF}$. As in the original signcryption schemes, the basic idea, roughly, is that, similar to the ElGamal encryption scheme [4], the sender Alice chooses a random 'session secret' and uses the resulting Diffie-Hellman shared key with the recipient Bob to encrypt the message. But instead of simply appending the public exponential of the 'session secret' to the ciphertext (with a signature on the message performed separately), the public exponential is used also as the 'commitment' for a Schnorr-like signature scheme. The 'commitment' is recoverable by the recipient Bob from the signature (saving the communication overhead of appending it in the ElGamal scheme), and with a modification to the signature part, the sender Alice need not even compute the 'commitment' explicitly, allowing a saving of computation.

More precisely, the signcryption scheme $\mathcal{SCF}$ is defined as follows. In the following, when we say X 'large' we mean that $log(X) = b \cdot k$ where b is a positive constant and k is the security parameter.

To set up a cryptographic 'community' using the scheme $\mathcal{SCF}$, a trusted authority generates and publishes three parameters:

Common Parameters Published by Trusted Authority

(1) A large RSA modulus $N = pq$ with p and q random primes of approximately equal length.
(2) A large secret-key bound $S \ll \sqrt{N}$.
(3) An element $g \in \mathbb{Z}_N^*$ of large order $1 \ll \mathrm{Ord}_{\mathbb{Z}_N^*}(g) \leq S/2$, which is an asymmetric basis in $\mathbb{Z}_N^*$, i.e. the multiplicity of 2 in $\mathrm{Ord}_{\mathbb{Z}_p^*}(g)$ is not equal to the multiplicity of 2 in $\mathrm{Ord}_{\mathbb{Z}_q^*}(g)$.

We note that the element g is easy to generate by a trusted authority if it generates p and q appropriately — A recommended common parameter generation algorithm is outlined in Sect. 4. We remark that Pointcheval's [11] definition of an asymmetric basis in $\mathbb{Z}_N^*$ is a special case of our more relaxed definition.

Then user Alice generates a secret/public key-pair as follows:

Key-pair generation by user Alice

(1) Alice picks her secret key integer s_A randomly and uniformly in the in the interval $\{0, ..., S-1\}$.
(2) Alice computes and publishes her public key $v_A = g^{-s_A} \bmod N$.

User Bob generates his key-pair s_B and v_B in the same way.

When Alice wishes to send Bob a confidential message m such that Bob can verify that Alice is its originator, Alice follows the following steps:

Signcryption of m by Alice the Sender

Step A.1 Alice picks uniformly at random an element r from the set of integers $\{0, ..., R-1\}$, where R is such that $2^{k'} \stackrel{\text{def}}{=} R/(2^{|\mathsf{KH}|}S)$ is large.

Step A.2 Alice obtains a trusted copy of Bob's public key, computes $x = v_B^r \bmod N$ and then 'splits' this into the pair $(x_2, x_1) = \mathsf{H}_1(x)$, where $\mathsf{H}_1(.)$ denotes a 'one-way' hash function, which maps arbitrarily long inputs into a string of length $|\mathsf{H}_1|$ bits.

Step A.3 Alice uses a secure symmetric encryption algorithm E (with matching decryption algorithm D) to encrypt m using key x_1 to obtain the ciphertext $c = \mathsf{E}(x_1, m)$.

Step A.4 Alice uses her secret key to compute the pair (e, y) defined by $e = \mathsf{KH}(x_2, m, bind)$ and $y = r + e \cdot s_A$ (note absence of modular reduction), where $\mathsf{KH}(.,.)$ denotes a 'keyed' hash function of output length $|\mathsf{KH}|$ bits, with first argument being the key. In *bind* Alice inserts her own and Bob's public keys.

Step A.5 Alice sends the 'signcryptext' triple (c, e, y) to Bob.

The recipient Bob receives (c', e', y') (which may not be equal to (c, e, y) due to modification by an attacker), and follows the following steps:

UnSigncryption of (c', e', y') by Bob the Recipient

Step B.1 Bob uses a trusted copy of Alice's public key and his own secret key to compute $x' = (g^{y'} v_A^{e'})^{-s_B} \bmod N$ and then, using the same procedure followed by Alice to split x, Bob splits x' into $(x'_2, x'_1) = \mathsf{H}_1(x')$.

Step B.2 Bob decrypts c' using the symmetric key x'_1 into $m' = \mathsf{D}(x'_1, c')$.

Step B.3 Bob accepts message m' as being originated from Alice if and only if $e' = \mathsf{KH}(x'_2, m', bind)$.

We note that a simple way for Alice to perform the split at **(Step A.2)** is to set $x_1 = \mathsf{H}_1^{\mathsf{L}}(x)$ and $x_2 = \mathsf{H}_1^{\mathsf{U}}(x)$, where $\mathsf{H}_1^{\mathsf{L}}(x)$ and $\mathsf{H}_1^{\mathsf{U}}(x)$ denote the $|\mathsf{H}_1^{\mathsf{L}}|$ least- and $|\mathsf{H}_1^{\mathsf{U}}|$ most- significant bits of $\mathsf{H}_1(x)$ respectively, with $|\mathsf{H}_1^{\mathsf{L}}| + |\mathsf{H}_1^{\mathsf{U}}| = |\mathsf{H}_1|$ and both $2^{|\mathsf{H}_1^{\mathsf{L}}|}$ and $2^{|\mathsf{H}_1^{\mathsf{U}}|}$ are large. Also in practice one can implement KH using a one-way hash function $\mathsf{H}_2(.)$ by concatanating the key x_2 and the input m and then hashing the pair (x_2, m). The *bind* information containing Bob's public key prevents a signcryptext sent by Alice to Bob from being transformed into a valid signcryptext carrying the same message from Alice to a third user colluding with Bob (see [17] for details on this 'double-payment' problem).

4 Efficiency and Common Parameter Generation

4.1 Communication Overhead

The communication overhead of our scheme (where $|x| = \lceil \log_2(x) \rceil$ denotes the bit length of x) is

$$Comm_{\mathcal{SCF}} = |e| + |y| = |\mathsf{KH}(.)| + |R| = 2|\mathsf{KH}(.)| + |S| + k' \qquad (1)$$

compared with the communication overhead of the original signcryption scheme SCS1 [17]:

$$Comm_{SCS} = |\mathsf{KH}(.)| + |q| \tag{2}$$

and with the RSA sign-then-encrypt method, assuming Alice and Bob have moduli N_A and N_B respectively,

$$Comm_{RSA} = |N_A| + |N_B| \tag{3}$$

4.2 Computational Cost

The on-line computational cost of our signcryption algorithm S is dominated by the integer multiplication $e \cdot s_A$ while its off-line computational costs is dominated by the modular exponentiation $g^r \bmod N$. The number of bit operations for these computations can be estimated as follows. For these estimates, we assume the well-known square-and-multiply exponentiation algorithm and classical arithmetic for multiplication (cost of computing $x \cdot y$ is approximately $|x||y|$) and modular reduction (cost of computing $x \bmod y$ is approximately $(|x|-|y|)|y|$ when $|x| > |y|$).

$$Comp_{\mathcal{SCF},\text{Online } \mathsf{S}} = |e| \cdot |s_A| = |\mathsf{KH}(.)||S| \tag{4}$$

and

$$Comp_{\mathcal{SCF},\text{Offline } \mathsf{S}} = 1.5|R|(2|N|^2) = 3(|\mathsf{KH}(.)| + |S| + k')|N|^2 \tag{5}$$

For comparison, the signcryption scheme SCS1 has computational costs

$$Comp_{SCS1,\text{Online } \mathsf{S}} = |e| \cdot |s_A| + (|e| + |s_A| - |q|)|q| = 2|\mathsf{KH}(.)||q| \tag{6}$$

and

$$Comp_{SCS1,\text{Offline } \mathsf{S}} = 3|q||p|^2. \tag{7}$$

In the RSA 'sign-then-encrypt' technique, the signing exponentiation (to Alice's secret key) of the hashed message must be peformed online. We assume an efficient RSA scheme, in which small public exponents are used (so we neglect the computational cost of the encryption exponentiation to Bob's public exponent) and the Chinese Remainder Theorem (CRT) is used to perform the exponentiation to Alice's secret key (even though this forces Alice to store the secret prime factors of her modulus):

$$Comp_{RSA,\text{Online, } \mathsf{S}} = (3/4) \cdot |N_A|^3 \tag{8}$$

The unsigncryption operation is dominated by the two exponentiations in the computation of $x' = (g^{y'} v_A^{e'})^{-s_B} \bmod N$. Using a simple generalization of the square-and-multiply algorithm as suggested by Shamir (see full paper version of [17]), one can compute a product of two exponentials $g_1^{\beta_1} g_2^{\beta_2} \bmod N$ using on average $1.5(|\beta_1| - |\beta_2|) + 1.75|\beta_2|$ multiplications modulo N (assuming without

loss of generality that $|\beta_1| > |\beta_2|$). In the case of our unsigncryption operation, the two exponents are $\beta_1 = y' \cdot s_B$ and $\beta_2 = e' \cdot s_B$ with lengths $|\beta_1| = 2|S| + |\mathsf{KH}| + k'$ and $|\beta_2| = |\mathsf{KH}| + |S|$, so the average bit operation cost of the unsigncryption of $\mathcal{SCF}$ is estimated as:

$$Comp_{\mathcal{SCF},\mathsf{U}} = (1.5(|S| + k') + 1.75(|S| + |\mathsf{KH}|))2|N|^2 \tag{9}$$

This should be compared with the cost of unsigncryption for SCS1 (where both exponents can be reduced mod q prior to performing the exponentiation):

$$Comp_{SCS1,\mathsf{U}} = 1.75|q|(2|p|^2) \tag{10}$$

The computational cost of the 'decrypt-then-verify' RSA algorithm is identical to that of 'sign-then-encrypt' (except that Bob's secret key / modulus are used to decrypt).

4.3 Choice of Parameters

As we saw above, efficiency considerations encourage the use of a relatively small $S \ll \sqrt{N}$ and N. However, our scheme is trivially breakable by solving $CDLP(N, g, S, v_A)$ to extract a secret key from the public one using, for example, Shank's 'Baby-Step, Giant Step' algorithm, which takes about $\sqrt{S}$ multiplications in $\mathbb{Z}_N^*$. Therefore, S can be chosen such that a breaking time $T_{CDLP} = \sqrt{S}T_{MULT}(|N|)$ (where $T_{MULT}(|N|)$ estimates the time to perform a multiplication in $\mathbb{Z}_N^*$) represents a sufficient security level. Our security analysis suggests that there exist no significantly faster attacks, as long as the time T_{FACT} required to factor N given (N, g, S) is not less than T_{CDLP}. But when $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$ is composed of large prime factors, we know of no algorithm more efficient than $CDLP$ for factoring N which makes use of (g, S). Therefore, we can choose $|N|$ and $|S|$ so that the time $T_{NFS}(|N|)$ required to factor N using the fastest known algorithm for the standard factorization problem (namely Number Field Sieve) is not less than T_{CDLP}.

Tables 1 and 2 compare the efficiency of our scheme with the earlier signcryption scheme SCS1 and RSA 'sign-then-encrypt' with comparable security level. The assumptions made in constructing the table are the following:

(1) Selection of $|N|$ and $|S|$: Consistent with the discussion above, we chose $|S|$ and $|N|$ such that $T_{CDLP}(|S|, |N|) = T_{NFS}(|N|)$. We used the assumptions of Lenstra and Verheul [10] to estimate the functions T_{NFS} and T_{CDLP}. In particular, we took $T_{NFS} = c \cdot L(2^{|N|})$, where $L(N) = \exp(1.923 \cdot \ln(N)^{1/3} \cdot \ln(\ln(N))^{2/3})$ denotes the well-known heuristic running time estimate for NFS and the constant c fixed by the 10^4 MIPS-Years (MY) effort taken recently to factor the 512-bit modulus RSA-155 [3]. We took $T_{CDLP}(|N|, |S|) = b \cdot 2^{|S|/2}|N|^2$, with the constant b fixed by the $(2.2 \cdot 10^6 \cdot \sqrt{2}/9)$ MY effort estimated for $T_{CDLP}(|N|, |S|)$ with $|N| = |S| = 109$ bit (we refer the reader to [10] for more details). Since the signcryption scheme SCS1 can be attacked using a DLP variant of the NFS algorithm in $\mathbb{Z}_p^*$ with time estimate $T_{NFS}(|p|)$, or alternatively, using Shank's

baby-step giant step algorithm in $\mathbb{Z}_q^*$ in time $\sqrt{q}$, we set for comparable security level between the schemes SCS1 and $\mathcal{SCF}$, $|q| = |S|$ and $|p| = |N|$. Similarly, we assume the RSA modulus length $|N_A| = |N_B|$ is equal to $|N|$ in $\mathcal{SCF}$.

(2) Selection of $|\mathsf{KH}(.)|$: Given a signcryptext (c, e, y) from Alice to Bob, Bob can create an existential forgery (c', e, y) of a signcryptext from Alice to Bob by finding a new message m' such that $\mathsf{KH}((g^y v_A^e)^{-s_B}, m', bind) = e$. If $\mathsf{KH}(.)$ is a collision-resistant hash function, Bob would be expected to invest about $2^{|\mathsf{KH}(.)|}$ operations to complete this attack. Therefore, to be consistent with the security level defined by the choice of $|S|$ and $|N|$, we assume $|\mathsf{KH}(.)| = |S|/2$.

(3) Selection of k': From the statement of Theorem 1 in Sect. 5.3, one can see that the lower bound on the factoring algorithm success probability is significant only when $2^{k'}$ is much greater than the number of queries an active attacker is allowed to make to the signcryption algorithm. Since the number of queries an active attacker can make is normally much lower than the number of offline operations available to him, we assume that a choice of $k' = |\mathsf{KH}(.)|$ will satisfy the above requirement.

Of particular note is the efficiency tradeoff offered by our scheme compared with the earlier signcryption scheme, namely the on-line signcryption computational cost has been cut by about half due to the absence of the modular reduction, but this at the expense of an approximate doubling in the communication overhead, and an increase in the unsigncryption computational cost for the same reason.

security parameters			$\mathcal{SCF}$ Comm.	Comm. Overhead	Comm. Overhead
$\lvert N\rvert = \lvert p\rvert$	$\lvert S\rvert = \lvert q\rvert$	$\lvert\mathsf{KH}(.)\rvert = k'$	overhead(bits)	RSA / $\mathcal{SCF}$	SCS1 / $\mathcal{SCF}$
1024	132	66	329	6.2	0.6
2048	188	94	469	8.7	0.6
4096	263	131	657	12.5	0.6
8192	363	181	907	18.1	0.6
10240	402	201	1004	20.4	0.6

Table 1. Comparison of Communication overhead of proposed signcryption scheme $\mathcal{SCF}$ with RSA based Signature-Then-Encryption with *Small Public Exponents* and CRT decryption and with original signcryption scheme SCS1.

4.4 Common Parameter Generation

Our unforgeability security proof in Sect. 5.3 requires that the common parameters satisfy (1) $N = p \cdot q$ for p and q prime, (2) $\mathrm{Ord}_{\mathbb{Z}_N^*}(g) < S/2$, (3) g is an asymmetric basis in $\mathbb{Z}_N^*$, and (4) Factoring N given (g, N, S) is intractable. We cannot prove that a practical common parameter generation algorithm $\mathsf{GenComm}$ exists which satisfies property (4). Based on known attacks on the standard factorization problem and $CDLP$, however, we conjecture that imposing the fol-

security parameters			Online	Online	Total	Total
$\lvert N\rvert = \lvert p\rvert$	$\lvert S\rvert = \lvert q\rvert$	$\lvert\mathsf{KH}(.)\rvert = k'$	RSA / $\mathcal{SCF}$	SCS1 / $\mathcal{SCF}$	RSA / $\mathcal{SCF}$	SCS1 / $\mathcal{SCF}$
1024	132	66	4.7E+04	2.0	0.74	0.41
2048	188	94	1.8E+05	2.0	1.04	0.41
4096	263	131	7.5E+05	2.0	1.48	0.41
8192	363	181	3.1E+06	2.0	2.15	0.41
10240	402	201	5.0E+06	2.0	2.43	0.41

Table 2. Ratio comparison of Online (by sender) and Total (by sender and receiver) computation cost of proposed signcryption scheme $\mathcal{SCF}$ with RSA based Signature-Then-Encryption (using *Small Public Exponents* and CRT decryption) and with original signcryption scheme SCS1.

lowing additional properties will allow (4) to be met: (5) $p-1$ and $q-1$ are not smooth (i.e. have at least one large prime factor) and (6) $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$ is large and composed of large prime factors (besides an unavoidable factor of 2 due to asymmetricity requirement). We therefore believe that the trusted authority can satisfy all the properties (1) to (6) using the following practical implementation of $\mathsf{GenComm}$: (1) Pick distinct random primes r_p and r_q each of length $(|S|-6)/2$ (2) Pick random odd t_p of length $|N|/2-|r_p|$ until $p = t_p r_p + 1$ passes a primality test. (3) Pick random odd t_q of length $|N|/2 - |r_q| - 1$ until $q = 2t_q r_q + 1$ passes a primality test (4) Compute $N = pq$ (5) Pick random $h_p \in \mathbb{Z}_p^*$ until $g_p = h_p^{(p-1)/r_p} \bmod p \neq 1$ (6) Pick random $h_q \in \mathbb{Z}_q^*$ (with $h_q \bmod q \neq q-1$) until $h_q^{(q-1)/2} \bmod q \neq 1$ and $g_q = h_q^{(q-1)/2r_q} \bmod q \neq 1$ (7) Using CRT compute $g = g_p q(q^{-1} \bmod p) + g_q p(p^{-1} \bmod q) \bmod N$. Note that the resulting g has order

$$\mathrm{Ord}_{\mathbb{Z}_N^*}(g) = \mathrm{lcm}(\mathrm{Ord}_{\mathbb{Z}_p^*}(g), \mathrm{Ord}_{\mathbb{Z}_q^*}(g)) = \mathrm{lcm}(r_p, 2r_q) = 2r_p r_q \qquad (11)$$

Finally, we remark that the requirement of our scheme to share among all users a set of common parameters defining a specific group, is similar to the recommendation in several recent standards (see [7] and [16]) of specific elliptic curves and group parameters, for the implementation of elliptic curve cryptographic algorithms.

4.5 Trusted Authority

As can be seen in Sect. 4.4, the common parameter generation algorithm for generating g requires knowledge of the factors (p, q) of N and large factors of $(p-1)$ and $(q-1)$. Thus the 'Trusted Authority' (TA) running the generation algorithm must be trusted to not make use of this knowledge to attack the system. But we emphasize that:

(1) Once (g, N, S) have been generated, our scheme has no further need for a TA (besides a public key certification authority, which is needed for any public key scheme), and

(2) The TA need not manipulate any user secret keys, since users can generate keys by themselves.

Hence, the TA for our scheme can cease to exist after (g, N, S) have been generated. The simplest form of TA is a sealed 'black box' device implementing the algorithm of Sect. 4.4 and programmed to erase from memory all traces of (p, q) (and the factors of $p-1$ and $q-1$) after g and N have been generated. Alternatively, the TA can take the form of a *group* of users who engage in a 'private' distributed common parameter generation algorithm, which prevents a minority of colluding dishonest participants from gaining knowledge on (p, q). We have in mind an efficient protocol similar to that recently proposed by Boneh and Franklin [2], although the present case is more challenging due to the requirements on g.

4.6 Trading Efficiency for Security

We observe that by choosing g of maximal order $\lambda(N)$, and $S \gg \sqrt{N}$ (eg $|S| > 0.6|N|$) one can use Poupard and Stern's simulation proof technique [13] to prove the unforgeability of our scheme relative to $CDLP$ to base g, which is harder than the *standard* factorization problem. This is due to the following two reasons: (1) Poupard and Stern's reduction proof assumes that, given (g, N, S), one can select an integer x uniformly from a certain set X such that with overwhelming probability, $g^x = g^s \bmod N$ for some $s \in \{0, \ldots, S-1\}$ (i.e. x maps to a valid public key) and there exists $y \in X$ such that $y \neq x$ and $g^x = g^y$ (i.e. given $g^x \bmod N$ an algorithm cannot determine with probability greater than $1/2$ the chosen value x). However, in contrast to our case, it appears that this assumption cannot be met unless $S \gg \sqrt{N}$. When the latter holds, one can approximate a multiple of $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$, namely $\phi(N) = N-(p+q)+1$ within the uncertainty in p and q, which is in the order of $\sqrt{N}$. So one can choose for the set X above the union of the *known* intervals $[0, S-1]$ and $[N, N+S-1]$ and achieve the desired properties above. (2) An element g of maximal order $\lambda(N)$ can be generated with relatively high probability of at least $1/[(7\ln\ln p)\cdot(7\ln\ln q)]$ (see [14]), given only $N = pq$, by simply picking g randomly in $\mathbb{Z}_N^*$. In this way, one can trade off the efficiency of our scheme in return for a proof of unforgeability relative to the standard factorization problem.

5 Security Analysis

5.1 Preliminaries

We use the notation $\mathsf{A}(.,.)$ to denote an algorithm, with input arguments separated by commas (our underlying computational model is a Turing Machine). If algorithm A makes calls to oracles, we list the oracles separated from the algorithm inputs by the symbol '|'. When discussing the program of an algorithm F which makes calls to an algorithm A, which in turn has oracle access to algorithm H, we use $Q^{(k)}_{\mathsf{A}-\mathsf{H}}[i]$ to denote the i'th query of A to H and $\rho_{\mathsf{A}-\mathsf{H}}[i]$ to denote

the corresponding i'th answer from H back to A, where these queries are understood to have been made during the execution of A called by the step labelled k in the program for algorithm F (if there is no ambiguity in the execution of A referred to, the superscript k is omitted). If an algorithm A is probabilistic (i.e. has a random input), we use the same notation as above to denote it with the understanding that it is invoked with a random input which is statistically independent of its outcomes in previous runs. In some cases, it will be useful to invoke A with its random input set to a specified value - in these cases we write $\mathsf{A}(.; w)$ to denote the deterministic algorithm obtained by running A with its random input set to w. Let $T(k)$ denote an upper bound on the number of computation steps performed by algorithm A with input parameter k. We say A has a *polynomial time bound* in k if there exists $c \in \mathbb{R}$, and $k_0 \in \mathbb{N}$ such that $T(k) \leq k^c$ for all $k > k_0$. We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if, for each $c \in \mathbb{R}$, there exists a k_0 such that $f(k) < 1/k^c$ for all $k > k_0$. We call a probability function $f : \mathbb{N} \rightarrow [0, 1]$ *overwhelming* if the function $g : \mathbb{N} \rightarrow [0, 1]$ defined by $g(k) = 1 - f(k)$ is negligible.

5.2 Security Notions for Signcryption Schemes

In this section, we adapt the 'asymptotic' digital signature security notions defined in [9] to signcryption schemes. Our presentation style is similar to that of Bellare (see, eg [1]), since we believe it makes all the security proof assumptions explicit. In order to define precise notions of security for a signcryption scheme, we need first a precise definition of a signcryption scheme itself.

Definition 1. *A Signcryption Scheme* $\mathcal{SCR} = (\mathsf{GenComm}, \mathsf{GenUser}, \mathsf{S}, \mathsf{U})$ *is an ordered sequence of four algorithms:*

1. A probabilistic common parameter/oracle generation algorithm $\mathsf{GenComm}$*, which takes as input a security parameter* k *and returns a pair* $(CommPar, \mathcal{O})$*, where* $CommPar$ *is a sequence of common parameters and* $\mathcal{O} = (\mathsf{H}_1, \mathsf{H}_2, ..., \mathsf{H}_{|\mathcal{O}|})$ *is a a sequence of* $|\mathcal{O}|$ *oracles* $\mathcal{O}[i] = \mathsf{H_i} : \{0,1\}^{L_{in}[i]} \rightarrow \{0,1\}^{L_{out}[i]}$.

2. A probabilistic user key-pair generation algorithm $\mathsf{GenUser}$*, which takes as input a security parameter* k *and a common parameter sequence* $CommPar$ *and returns a single user's secret/public key-pair* (sk, pk).

3. A probabilistic Signcryption algorithm S*, which takes as input a security parameter* k*, a common parameters sequence* $CommPar$*, a sender's secret key* sk_A*, a recipient's public key* pk_B*, and a message* $m \in M$ *(*M *is the message space), has access to oracles in a sequence* $\mathcal{O}$ *and returns a signcryptext* C_S.

4. An UnSigncryption algorithm U*, which takes as input a security parameter* k*, a common parameters sequence* $CommPar$*, a recipient's secret key* sk_B*, a sender's public key* pk_A*, a signcryptext* C_S*, and has access to oracles in a sequence* $\mathcal{O}$ *and returns a pair* (m, b)*, consisting of a message* m *and a verification bit* b.

The translation of the informal definition of the scheme $\mathcal{SCF}$ in section 3 is straightforward. We simply highlight that our proof of security applies to the

following version of our scheme: (1) The keyed-hash $\mathsf{KH}(.,.)$ is implemented using concatanation and the one-way hash function $\mathsf{H}_2(.)$ (2) The splitting of x into x_1 and x_2 is into the least significant and most significant bits of $\mathsf{H}_1(x)$ denoted $\mathsf{H}_1^{\mathsf{L}}(x)$ and $\mathsf{H}_1^{\mathsf{U}}(x)$ respectively (3) The algorithm $\mathsf{GenComm}$ outputs $CommPar = (g, N, S)$ and $\mathcal{O} = (\mathsf{H}_1, \mathsf{H}_2, \mathsf{E}, \mathsf{D})$ (4) We shall assume the following property of $\mathsf{GenComm}$: The outputs $CommPar$ and $\mathcal{O}$ are *statistically independent* (this includes as a special case the most practical version where the hash and encryption algorithms in $\mathcal{O}$ are derived deterministically from k). This assumption simplifies the intractability assumption required in the security proof.

Apart from security considerations, a necessary condition for a signcryption scheme to be usable is that it must 'work' as expected in the absence of attackers. Such a scheme is called complete.

Definition 2. *Define the experiment*

Experiment CompExp$(\mathcal{SCR}, k, m)$
$(CommPar, \mathcal{O}) \leftarrow \mathsf{GenComm}(k)$
$(sk_A, pk_A) \leftarrow \mathsf{GenUser}(k, CommPar)$
$(sk_B, pk_B) \leftarrow \mathsf{GenUser}(k, CommPar)$
$C_S \leftarrow \mathsf{S}(k, CommPar, sk_A, pk_B, m|\mathcal{O})$
$(m', b) \leftarrow \mathsf{U}(k, CommPar, sk_B, pk_A, C_S|\mathcal{O})$
If $m' = m$ and $b = 1$ then **Return** 1 else **Return** 0

We say that signcryption scheme $\mathcal{SCR}$ is complete *if, for all messages m,* $\Pr[\mathbf{CompExp}(\mathcal{SCR}, k, m) = 1]$ *is an overwhelming probability function in k.*

The completeness of our scheme $\mathcal{SCF}$ follows from a simple calculation, namely, using the notation of section 3 (with all arithmetic performed $\bmod N$), if $(c', e', y') = (c, e, y)$ then:

$$x' = (g^{y'} v_A^{e'})^{-s_B} = ((g^{r+s_A \cdot e} g^{-s_A \cdot e})^{-s_B} = g^{r \cdot (-s_B)} = v_B^r = x \qquad (12)$$

and therefore $(x_1', x_2', m') = (x_1, x_2, m)$, so the message m is recovered and the verification test is passed (by convention, the unsigncryption algorithm U returns a verification bit equal to 1 when the test is passed).

Authenticity security properties for signcryption schemes can be classified similarly to signature schemes, according to the resources available to the attacker and severity of the attacker's output forgery (see [9] and [12]). We shall take the most conservative security notion — we will call a signcryption scheme unforgeable only if it is infeasible for the most resourceful type of attacker (namely an 'adaptively chosen message attacker') to produce the weakest type of forgery (namely an 'existential' forgery).

Definition 3. *Let $\mathcal{SCR} = (\mathsf{GenComm}, \mathsf{GenUser}, \mathsf{S}, \mathsf{U})$ be a signcryption scheme using $|\mathcal{O}|$ oracles. Let $R \subseteq \{1, ..., |\mathcal{O}|\}$. Define the experiment*

Experiment ForgeExp $(k, \mathcal{SCR}, \mathsf{A}, R)$
$(CommPar, \mathcal{O}) \leftarrow \mathsf{GenComm}(k)$
$(sk_A, pk_A) \leftarrow \mathsf{GenUser}(k, CommPar)$
$(sk_B, pk_B) \leftarrow \mathsf{GenUser}(k, CommPar)$
For each $i \in \{1, ..., |\mathcal{O}|\}$

If $i \in R$ Pick a random function $\mathsf{H}'_i : \{0,1\}^{L_{in}[i]} \rightarrow \{0,1\}^{L_{out}[i]}$
Else $\mathsf{H}'_i \leftarrow \mathcal{O}[i]$
$\mathcal{O}' \leftarrow (H'_1, ..., H'_{|\mathcal{O}|})$
$C_S \leftarrow \mathsf{A}(k, CommPar, pk_A, pk_B, sk_B | \mathcal{O}', \mathsf{S}(k, CommPar, sk_A, pk_B, . | \mathcal{O}'))$
$(m, b) \leftarrow \mathsf{U}(k, CommPar, sk_B, pk_A, C_S | \mathcal{O}')$
If $b = 1$ and $m \neq Q_{\mathsf{A}-\mathsf{S}}[i]$ for all i then **Return** 1
Else **Return** 0

We say that signcryption scheme $\mathcal{SCR}$ is existentially unforgeable under an adaptively chosen message attack with respect to random oracle replacement set R if, for any algorithm A with polynomial time bound in k, $\Pr[\mathbf{ForgeExp}(k, \mathcal{SCR}, \mathsf{A}, R) = 1]$ *is a negligible function in k.*

In Definition 3, the set R specifies the (indexes of) oracles in the sequence $\mathcal{O}$ which are to be replaced by randomly chosen functions in the random oracle model of the signcryption scheme $\mathcal{SCR}$, to yield the oracle array $\mathcal{O}'$. The attacker A's success probability in yielding a valid forgery is then assessed over all random choices made in experiment **ForgeExp** including the choice of the random functions for inclusion in $\mathcal{O}'$.

5.3 Security Properties of New Signcryption Scheme

We first examine the authentication security of the signcryption scheme $\mathcal{SCF}$. We will prove (using the methods of Pointcheval [11]) the unforgeability of the scheme $\mathcal{SCF}$ by showing (in Theorem 1) how to use a poly-time attacker for $\mathcal{SCF}$, which produces a valid forgery with non-negligible probability, to construct a poly-time algorithm which factors the public modulus N with non-negligible probability. Then the unforgeability of $\mathcal{SCF}$ follows (in Corollary 1) from the intractability assumption (see Definition 4) on factorization of N given the triple (g, N, S) output by $\mathsf{GenComm}$.

There are two key ideas in performing the reduction from factorization to forgery of $\mathcal{SCF}$ in the proof of Theorem 1. The first is the use of the 'forking' technique [12] in extracting (from two executions of the attacker) two signcryptexts (c_1, e_1, y_1) and (c_2, e_2, y_2) with the same 'commitment' x^*, i.e. satisfying $(g^{y_1} v_A^{e_1})^{-s_B} = (g^{y_2} v_A^{e_2})^{-s_B}$. From these two forgeries, the factoring algorithm can then compute a multiple of $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$, namely $L \stackrel{\text{def}}{=} s_B[(y_2 - y_1) - s_A(e_2 - e_1)]$. Once L is known, it is then easy to factor N using the fact that g is an asymmetric basis in $\mathbb{Z}_N^*$ (see Lemma 6 in Appendix). Of course, L cannot be used to factor N if L is the trivial zero multiple of $\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$. Here we need the second key idea, namely the *Witness Indistinguishability* (WI) of the signcryption algorithm S, to show that $L \neq 0$ with high probability.

The WI property, first studied by Feige and Shamir [5], can hold non-trivially only for schemes in which the one-way function $f(.)$ transforming the secret key space to the public key space is not one-to-one. In fact, an even stronger requirement is needed, namely: For each public key v, the preimage $W(v) = \{s : f(s) = v\}$ of v under $f(.)$ contains at least $n \geq 2$ secret keys. In the case of our

scheme, we have $f(s) = g^{-s} \bmod N$ which can be made to map at least n secret keys to each public key by choosing the secret key space as $\{0, \ldots, S-1\}$ with $S \geq n \cdot \mathrm{Ord}_{\mathbb{Z}_N^*}(g)$. Under this assumption, the statistical WI property of the signcryption algorithm S (Lemma 3 in Appendix) means that the probability distribution of signcryptexts output by S is almost independent of which secret key in $W(v_A)$ is used by the sender having public key v_A.

Referring back to the reduction above, note that the undesirable outcome $L = 0$ results (apart from the negligibly probable case $s_B = 0$) from the outcome $h(e_1, e_2, y_1, y_2) \stackrel{\text{def}}{=} (y_2 - y_1)/(e_2 - e_1) = s_A$. It is straightforward to see that if the joint probability distribution of singcryptexts observed by the attacker is perfectly WI, i.e. completely independent of s_A in $W(v_A)$, then choosing s_A randomly in $\{0, 1, ..., S-1\}$ before running the attacker would give the undesirable outcome $h(e_1, e_2, y_1, y_2) = s_A$ with probability not greater than ϵ/n, where n is the number of secret keys in $W(v_A)$ and ϵ is the attacker's success probability in computing the multiple L. However, since the signcryption algorithm S is only *statistically* WI, we can only prove that the distribution of the signcryptexts observed by the attacker is *approximately* independent of s_A in $W(v_A)$, and we need to bound the effect of this on the probability of the outcome $L = 0$ to ensure that the factoring algorithm still succeeds with non-negligible probability even when the attacker queries S a polynomial number of times (our proof makes use of Lemma 2 in Appendix for this purpose).

Theorem 1. *Let* A *be an adaptively chosen message attacker algorithm for signcryption scheme* $\mathcal{SCF}$=($\mathsf{GenComm}$,$\mathsf{GenUser}$,S,U)*, having running time bound* T *and success probability*

$$\epsilon \stackrel{\text{def}}{=} \Pr[\mathbf{ForgeExp}(k, \mathcal{SCF}, \mathsf{A}, \{1,2\}) = 1], \tag{13}$$

where **ForgeExp** *is the experiment in Definition 3. Suppose that* $l_{\mathsf{A-S}}$, $l_{\mathsf{A-H_1}}$ *and* $l_{\mathsf{A-H_2}}$ *are upper bounds on the number of oracle queries* A *makes to* S, $\mathsf{H_1}$ *and* $\mathsf{H_2}$*, respectively. Let* S_L *denote a lower bound on* S *output by* $\mathsf{GenComm}$*. Define the experiment*

Experiment FactorExp$(k, \mathsf{GenComm}, \mathsf{Fact})$
$(g, N, S, \mathcal{O}) \leftarrow \mathsf{GenComm}(k)$
$p \leftarrow \mathsf{Fact}(k, g, N, S)$
If p is a non-trivial divisor of N then **Return** 1
Else **Return** 0

There exists a factoring algorithm $\mathsf{Fact}(.,.,.,.)$ *with running time bound*

$$T' = 2T + l_{\mathsf{A-S}} O(k^3) + O((l_{\mathsf{A-S}} + l_{\mathsf{A-H_2}})^2)$$

and success probability

$$\epsilon' = \Pr[\mathbf{FactorExp}(k, \mathsf{GenComm}, \mathsf{Fact}) = 1] \geq \epsilon_*(\epsilon_* - 8l_{\mathsf{A-S}}/2^{k'})/8 - 1/S_L,$$

where

$$\epsilon_* \stackrel{\text{def}}{=} (\epsilon - 1/2^{|\mathsf{H_2}|})^2/(16 l_{\mathsf{A-H_2}}) - (l_{\mathsf{A-H_1}} + 1)(2 l_{\mathsf{A-H_1}} + 1) \cdot (1/2^{|\mathsf{H_1^U}|}).$$

Proof. We present a factoring algorithm $\mathsf{Fact}(.,.,.,.)$ and then we analyse it to show that it has the claimed running time bound and success probability. Unless stated otherwise, all products involving g are performed in the multiplicative group $\mathbb{Z}_N^*$, where the multiplication operation is performed modulo N.

The following factoring algorithm makes calls to a 'modified' attacker algorithm A'. The modified attacker A' has precisely the same output probability distribution (and hence success probability) as the given attacker A, but A' has the following additional property: For all executions of A', each query which A' makes to oracle H_2 is *distinct* from all previous queries made in the execution to H_2 (by either A' itself or by the signcryption algorithm S). An arbitrary attacker A can always be modified into A' satisfying the above properties in the following way: A' operates in the same way as A but maintains a 'lookup table' of all queries previously made to oracle H_2 and the corresponding answers returned by H_2. Then whenever A makes a query Q to H_2, the modified attacker A' first searches its lookup table for a previous occurrence of Q, and if found, takes the answer from the table without querying H_2 (if not found, the query is made to H_2 as usual). In order to maintain the lookup table, A' must update it with a query/answer pair whenever a new query is made to H_2. This is trivial when the new query is made by A' directly. However, a new query to H_2 may also be made 'indirectly' when A' queries the signcryption algorithm S with a message m. This results in the query $(\mathsf{H}_1^{\mathsf{U}}(v_B^r), m, bind)$ from S to H_2 and the signcryptext (c, e, y) is returned to A'. In order to update its table, A' recovers $v_B^r = (g^y v_A^e)^{-s_B}$ and queries H_1 to recover $\mathsf{H}_1^{\mathsf{U}}(v_B^r)$. Then it has the query/answer pair $[(\mathsf{H}_1^{\mathsf{U}}(v_B^r), m, bind), e]$ and updates the table. We note that maintaining the lookup table can be done in time $O((l_{\mathsf{A-S}} + l_{\mathsf{A-H_2}})^2)$, plus a time $l_{\mathsf{A-S}}O(k^3)$ to recover S's queries as described above.

Algorithm $\mathsf{Fact}(k, g, N, S)$

1. $(v_A, s_A) \leftarrow \mathsf{GenUser}(k, g, N, S)$
2. $(v_B, s_B) \leftarrow \mathsf{GenUser}(k, g, N, S)$
3. $(g', N', S', (\mathsf{H}_1, \mathsf{H}_2, \mathsf{E}, \mathsf{D})) \leftarrow \mathsf{GenComm}(k)$
4. Pick random functions $\mathsf{H}_1' : \{0,1\}^* \rightarrow \{0, ..., 2^{|\mathsf{H}_1|} - 1\}$ and $\mathsf{H}_2' : \{0,1\}^* \rightarrow \{0, ..., 2^{|\mathsf{H}_2|} - 1\}$ and form $\mathcal{O}' = (\mathsf{H}_1', \mathsf{H}_2', \mathsf{E}, \mathsf{D})$.
5. Pick random inputs for running A':
   (a) Pick random $\omega_{A'} \in \{0,1\}^*$.
   (b) Pick l_S random elements $r'[i] \in \{0, ..., R-1\}$ for $1 \leq i \leq l_{\mathsf{A-S}}$.
6. $(e_1, y_1, c_1) \leftarrow \mathsf{A}'(k, g, N, v_A, v_B, s_B; \omega_{A'} | \mathsf{S}(k, g, N, s_A, v_B, .; r'[i] | \mathcal{O}'), \mathcal{O}')$
7. $(m_1, b_1) \leftarrow \mathsf{U}(k, g, N, v_A, s_B, (e_1, y_1, c_1) | \mathcal{O}')$
8. $t_1 \leftarrow \mathsf{Find}(Q^{(6)}_{\mathsf{A'-S}}, m_1, l_{\mathsf{A-S}})$
9. $t_2 \leftarrow \mathsf{Find}(Q^{(6)}_{\mathsf{A'-H_2'}}, (\mathsf{H}_1'^{\mathsf{U}}((g^{y_1} v_A^{e_1})^{-s_B}), m_1, bind), l_{\mathsf{A-H_2}})$
10. If $b_1 \neq 1$ or $t_1 \neq 0$ then **Abort**
11. If $t_2 = 0$ then **Abort**
12. Pick a random function $\mathsf{H}_2'' : \{0,1\}^* \rightarrow \{0, ..., 2^{|\mathsf{H}_2|} - 1\}$ subject to the restriction $\mathsf{H}_2''(Q^{(6)}_{\mathsf{A'-H_2'}}[i]) = \mathsf{H}_2'(Q^{(6)}_{\mathsf{A'-H_2'}}[i])$ for all $1 \leq i \leq (t_2 - 1)$ and $\mathsf{H}_2''(Q^{(6)}_{\mathsf{S-H_2'}}[i]) = \mathsf{H}_2'(Q^{(6)}_{\mathsf{S-H_2'}}[i])$ for all $1 \leq i \leq j$, where j is the

number of queries S made to H'_2 before the query $Q_{\mathsf{A}'-\mathsf{H}'_2}[t_2]$ in the execution of step 6, and form $\mathcal{O}'' = (\mathsf{H}'_1, \mathsf{H}''_2, \mathsf{E}, \mathsf{D})$.

13. Pick $l_{\mathsf{A}-\mathsf{S}}$ random elements $r''[i] \in \{0, ..., R-1\}$ for $i \in \{1, ..., l_{\mathsf{A}-\mathsf{S}}\}$, subject to the restriction $r''[i] = r'[i]$ for all $i \in \{1, ..., j\}$.
14. $(e_2, y_2, c_2) \leftarrow \mathsf{A}'(k, g, N, v_A, v_B, s_B; \omega_{A'} | \mathsf{S}(k, g, N, s_A, v_B, .; r''[i] | \mathcal{O}''), \mathcal{O}'')$
15. $t'_1 \leftarrow \mathsf{Find}(Q^{(14)}_{\mathsf{A}'-\mathsf{S}}, m_2, l_{\mathsf{A}-\mathsf{S}})$
16. $t'_2 \leftarrow \mathsf{Find}(Q^{(14)}_{\mathsf{A}'-\mathsf{H}''_2}, (\mathsf{H}'^{\mathsf{U}}_1((g^{y_2} v_A^{e_2})^{-s_B}), m_2, bind), l_{\mathsf{A}-\mathsf{H}_2})$
17. $(m_2, b_2) \leftarrow \mathsf{U}(k, g, N, v_A, s_B, (e_2, y_2, c_2) | \mathcal{O}'')$
18. If $b_2 \neq 1$ or $t'_1 \neq 0$ or $t'_2 \neq t_2$ or $e_1 = e_2$ then **Abort**
19. If $(g^{y_1} v_A^{e_1})^{-s_B} \neq (g^{y_2} v_A^{e_2})^{-s_B}$ then **Abort**
20. $L \leftarrow s_B[(y_2 - y_1) - s_A(e_2 - e_1)]$
21. If $L = 0$ then **Abort**
22. $p \leftarrow \mathsf{MillFact}(g, L, N)$
23. **Return** p

By the shorthand notation in step 6 above we mean that Fact answers the i'th query $Q_{\mathsf{A}'-\mathsf{S}}[i]$ of A' to S by running S with its random input set to $r[i]$. Algorithm Find is defined as follows: $\mathsf{Find}(Q, m, l)$ searches array Q of size l for the element m, returning the index of m in Q if found, or else returning 0 if Q does not contain m. The algorithm $\mathsf{MillFact}$ is a variant of Miller's factoring algorithm (see Lemma 6 in Appendix).

We note first that if algorithm Fact is called by experiment **FactorExp** and completes execution without aborting, then it successfully factors N. This is because, when step 19 does not abort, we have $(g^{y_1} v_A^{e_1})^{-s_B} = (g^{y_2} v_A^{e_2})^{-s_B}$, which implies that $g^L = 1$ (with L defined in step 20), or equivalently (since step 21 does not abort), L is a non-zero multiple of $\mathrm{Ord}_{\mathbb{Z}^*_N}(g)(g)$. Hence, since g output by $\mathsf{GenComm}$ is an asymmetric basis in $\mathbb{Z}^*_N$, then by Lemma 6 in the Appendix, step 22 computes a non-trivial divisor of N in time $O(|L| \cdot |N|^2) = O(k^3)$ (assuming that the unsigncryption algorithm rejects a signcryptext (c, e, y) when y is outside the known range $[0, R + |\mathsf{H}_2| \cdot S]$, so that during a successful run of Fact, when both runs of the attacker output a successful forgery, both y_1 and y_2 are guaranteed to be in this range).

We now lower bound the probability that Fact does not abort. In the following analysis, we denote by $Ab(n)$ the event of executions in which Fact aborts at step n, and by $\neg Ab(n)$ its complement, Note that if the first abort test following step n occurs at step m, then $\neg Ab(n) = Ab(m) \cup \neg Ab(m)$. Consider first the abort test in step 10. Observe that the algorithm up to this point is just a single run of the forgery experiment **ForgeExp**, and $\neg Ab(10)$ corresponds to a successful forgery, i.e. **ForgeExp** $= 1$. Therefore $\Pr[\neg Ab(10)] = \epsilon$.

Now consider the abort test at step 11. The algorithm aborts at this step if the query $Q_{F1} \stackrel{\text{def}}{=} (\mathsf{H}'^{\mathsf{U}}_1((g^{y_1} v_A^{e_1})^{-s_B}), m_1, bind)$ (which we call the 'forgery query'), asked by U to H'_2 in step 7 to verify the attacker's output forgery (c_1, e_1, y_1) has not been previously asked by A' to H'_2. In that case the answer returned by H'_2 to U is uniform and independent of the attacker's output and we have

$\Pr[Ab(11)] = \Pr[\neg Ab(10) \cap t_2 = 0] \leq \Pr[b_1 = 1 \cap t_2 = 0] \leq 1/2^{|\mathsf{H}_2|}$. We deduce that $\epsilon_1 \stackrel{\text{def}}{=} \Pr[\neg Ab(11)] = \Pr[\neg Ab(10)] - \Pr[Ab(11)] \geq \epsilon - 1/2^{|\mathsf{H}_2|}$.

Next, we consider the abort test in step 18. Observe that the probability space Ω of execution outcomes of the attacker A' can be represented as a product space $\Omega_{a(i)} \times \Omega_{b(i)}$ in i different ways for each $i \in \{1, ..., l_{\mathsf{A}-\mathsf{H}_2}\}$. In these products, the space $\Omega_{a(i)}$ includes all random outcomes occurring *before* the i'th query of A' to H_2, and the space $\Omega_{a(i)}$ includes all remaining random outcomes occurring after that query. Below, we let $(a_i, b_i) \in \Omega_{a(i)} \times \Omega_{b(i)}$ denote the execution outcome of the first attacker run in step 6 and we let $(a'_i, b'_i) \in \Omega_{a(i)} \times \Omega_{b(i)}$ denote the outcome of the second attacker run in step 14. Define the event $S \subseteq \Omega$ consisting of runs of A' where a successful forgery is output (i.e. **ForgeExp**=1) and the 'forgery query' is asked to H_2. The event S can be partitioned into the $l_{\mathsf{A}-\mathsf{H}_2}$ subevents S_i according to the index i of the 'forgery query' during the run. Therefore, the event $\neg Ab(11) \cap (t_2 = i)$ in the execution of Fact corresponds to the event $(a_i, b_i) \in S_i$ for the first attacker run in step 6, and similarly, the event $(b_2 = 1 \cap t'_1 = 0 \cap t'_2 = i)$ for Fact corresponds to $(a'_i, b'_i) \in S_i$ for the second run of A' in step 14. Note that (i) The marginal probability distributions of (a_i, b_i) and (a'_i, b'_i) respectively are both equal to the random outcome distribution of the attacker run in Experiment **ForgeExp**, and (ii) If $(a_i, b_i) \in S_i$ then $a'_i = a_i$ because Fact performs the second attacker run with all random outcomes prior to the forgery query $Q_{\mathsf{A}-\mathsf{H}_2[i]}$ identical to those in the first attacker run, namely equal to a_i.

Now since $\sum_{i=1}^{l_{\mathsf{A}-\mathsf{H}_2}} \Pr[(a_i, b_i) \in S_i] = \Pr[(a_i, b_i) \in S] \geq \epsilon_1$, there exists a set of indices i such that all the subevents S_i corresponding to those indices have high probability and their union has most of the probability of S. In particular, define $G \stackrel{\text{def}}{=} \{i \in \{1, ..., l_{\mathsf{A}-\mathsf{H}_2}\} : \Pr[S_i] \geq \epsilon_1/(2l_{\mathsf{A}-\mathsf{H}_2})\}$. Then one can easily show that

$$\sum_{i \in G} \Pr[S_i] \geq \epsilon_1/2 \ . \tag{14}$$

Applying the splitting lemma (see Lemma 4 in Appendix) to each subset S_i on product PS $\Omega_{a(i)} \times \Omega_{b(i)}$ we have the existence of corresponding subsets $\Omega_i \subseteq \Omega_{a(i)}$ satisfying

$$\Pr[a_i \in \Omega_i | (a_i, b_i) \in S_i] \geq 1/2 \tag{15}$$

and

$$\Pr[(a'_i, b'_i) \in S_i | (a'_i = a)] \geq = \Pr[S_i]/2 \geq \epsilon_1/(4l_{\mathsf{A}-\mathsf{H}_2}) \text{ for all } a \in \Omega_i \ . \tag{16}$$

Now we can lower bound the probability of passing all abort tests up to and including step 18. Writing $\alpha_i \stackrel{\text{def}}{=} (a_i, b_i)$ and $\alpha'_i = (a'_i, b'_i)$, we have

$$\begin{aligned} &\Pr[\neg Ab(18)] \\ &= \sum_{i=1}^{l_{\mathsf{A}-\mathsf{H}_2}} \Pr[(\alpha_i \in S_i) \cap (\alpha'_i \in S_i) \cap (e_1 \neq e_2)] \end{aligned}$$

$$\geq \sum_{i \in G} \Pr[(\alpha_i \in S_i) \cap (a_i \in \Omega_i) \cap (\alpha'_i \in S_i) \cap (e_1 \neq e_2)]$$
$$= \sum_{i \in G} \sum_{\substack{a \in \Omega_i \\ (a,b) \in S_i}} \Pr[\alpha_i = (a,b)] \cdot \Pr[(\alpha'_i \in S_i) \cap (e_1 \neq e_2) | \alpha_i = (a,b)] \quad (17)$$

Note that

$$\Pr[(\alpha'_i \in S_i) \cap (e_1 \neq e_2) | \alpha_i = (a,b)]$$
$$= \Pr[\alpha'_i \in S_i | \alpha_i = (a,b)] - \Pr[(\alpha'_i \in S_i) \cap (e_1 = e_2) | \alpha_i = (a,b)] \quad (18)$$

Since A' never asks a previous query to H_2, the answer e_2 to the i'th query $Q'_{\mathsf{A}'-\mathsf{H}_2}$ in the second run is uniformly distributed in $\{0, ..., 2^{|H_2|} - 1\}$ independent of the corresponding answer e_1 in the first run. We therefore have, for the second term in (18):

$$\Pr[((a'_i, b'_i) \in S_i) \cap (e_1 = e_2) | (a_i\ , b_i) = (a,b)]$$
$$\leq \Pr[e_1 = e_2 | (a_i, b_i) = (a,b)] = 1/2^{|H_2|}. \quad (19)$$

For the first term in (18), we obtain, assuming $(a,b) \in S_i$ and $a \in \Omega_i$ and using (16),

$$\Pr[\alpha'_i \in S_i) | \alpha_i = (a,b)] = \Pr[\alpha'_i \in S_i) | a'_i = a] \quad (20)$$
$$\geq \epsilon_1/(4l_{\mathsf{A}-\mathsf{H}_2}). \quad (21)$$

Putting (19) and (20) into (18) we find:

$$\Pr[(\alpha'_i \in S_i) \cap (e_1 \neq e_2) | \alpha_i = (a,b)] \geq \epsilon_1/(4l_{\mathsf{A}-\mathsf{H}_2}) - 1/2^{|H_2|} \quad (22)$$

Putting (22) in (17), we find

$$\Pr[\neg Ab(18)] \geq \epsilon_1/(4l_{\mathsf{A}-\mathsf{H}_2}) \cdot \sum_{i \in G} \sum_{\substack{a \in \Omega_i \\ (a,b) \in S_i}} \Pr[\alpha_i = (a,b)]$$
$$= \epsilon_1/(4l_{\mathsf{A}-\mathsf{H}_2}) \cdot \sum_{i \in G} \Pr[a_i \in \Omega_i | \alpha_i \in S_i] \cdot \Pr[\alpha_i \in S_i]$$
$$\geq \epsilon_1/(4l_{\mathsf{A}-\mathsf{H}_2}) \cdot (1/2) \cdot \sum_{i \in G} \Pr[\alpha_i \in S_i] \text{ using (15)}$$
$$\geq \epsilon_1/(8l_{\mathsf{A}-\mathsf{H}_2}) \cdot (\epsilon_1/2) \text{ using (14)}$$
$$= \epsilon_1^2/(16l_{\mathsf{A}-\mathsf{H}_2}) \stackrel{\text{def}}{=} \epsilon_2. \quad (23)$$

Now note that (i) For executions in $\neg Ab(18)$, the verification query to H'_2 (or H''_2) is asked as the i'th query in both attacker runs, and (ii) The two attacker runs are identical up to (and including) the point where i'th query is asked to H'_2. Therefore, for those executions, the verification queries of the two attacker runs, namely $(\mathsf{H}'^{\mathsf{U}}_1((g_1^y v_A^{e_1})^{-s_B}), m_1, bind)$ and $(\mathsf{H}'^{\mathsf{U}}_1((g_2^y v_A^{e_2})^{-s_B}), m_2, bind)$

respectively, are identical. Therefore the two attacker runs can be considered together as a single algorithm $\mathsf{FindColl}$, which makes up to $2l_{\mathsf{A-H}_1}$ queries to the random function oracle $\mathsf{H}_1'^{\mathsf{U}}$, and outputs a collision pair of arguments for $\mathsf{H}_1'^{\mathsf{U}}$, namely $(x_1, x_2) \stackrel{\text{def}}{=} ((g^{y_1} v_A^{e_1})^{-s_B}, (g^{y_2} v_A^{e_2})^{-s_B})$ with probability at least ϵ_2. Applying Lemma 5 (see Appendix) to algorithm $\mathsf{FindColl}$, we see that the probability $\Pr[Ab(19)]$ that $x_1 \neq x_2$ cannot exceed $(2l_{\mathsf{A-H}_1}+2)(2l_{\mathsf{A-H}_1}+1)/2 \cdot (1/2^{|\mathsf{H}_1^{\mathsf{U}}|})$. Therefore,

$$\Pr[\neg Ab(19)] = \Pr[\neg Ab(18)] - \Pr[Ab(19)] \tag{24}$$

$$\geq \epsilon_2 - (l_{\mathsf{A-H}_1}+1)(2l_{\mathsf{A-H}_1}+1) \cdot (1/2^{|\mathsf{H}_1^{\mathsf{U}}|}) \stackrel{\text{def}}{=} \epsilon_3 \tag{25}$$

We will now use the WI of the scheme $\mathcal{SCF}$ to show that the last abort test is also passed with non-negligible probability. In the following, we let $T \stackrel{\text{def}}{=} \neg Ab(19)$, so that the executions in which Fact is successful form the event $T \cap \neg Ab(21)$. Also, we let $z \stackrel{\text{def}}{=} (y_2 - y_1)/(e_2 - e_1)$ if the execution is in T and set $z = xx$ if the execution is not in T (xx is just a symbol indicating that an abort has occurred before step (21)). We note the set equality $(T \cap Ab(21)) = (T \cap z = s_A) \cup (T \cap s_B = 0)$. Since $\Pr[(T \cap s_B = 0)] \leq \Pr[s_B = 0] = 1/S$ is negligible, we derive below a lower bound on $\Pr[T \cap (z \neq s_A))]$.

We begin by applying the Splitting Lemma (lemma 4 in Appendix) to the product probability space of common parameters/A secret key triples (g, N, s_A) by all the other random choices in Fact. Since $\Pr[T] \geq \epsilon_3$ over this product space, we deduce the existence of a subset Ω_T of triples (g, N, s_A) such that

$$\Pr[\Omega_T] \geq \epsilon_3/2 \tag{26}$$

and

$$\text{For each } (g', N', s'_A) \in \Omega_T, \Pr[T|(g, N, s_A) = (g', N', s'_A)] \geq \epsilon_3/2. \tag{27}$$

For each $(g', N', s'_A) \in \Omega_T$, let $v'_A \stackrel{\text{def}}{=} g^{-s_A} \bmod N'$ denote A's public key as usual, and let $W^{(g',N',v'_A)} \stackrel{\text{def}}{=} \{s \in \{0, ..., S-1\} : g'^{-s} \bmod N' = v'_A\}$ denote the set of secret keys of cardinality $|W^{(g',N',v'_A)}|$ which map to the public key v'_A. For brevity we let $\gamma \stackrel{\text{def}}{=} (g', N', v'_A)$. Denote by $s^{(\gamma)}(k)$ the k'th key in the set $W^{(\gamma)}$, indexed in an arbitrary manner. Now extend Ω_T to form $\Omega'_T \stackrel{\text{def}}{=} \{(g', N', s'_A) : \exists s^* \text{s.t.} s^* \in W^{(\gamma)} \text{and} (g', N', s^*) \in \Omega_T\}$, and $\Omega_T'^{(v)} \stackrel{\text{def}}{=} \{(\gamma) : \exists s' \text{s.t. } s' \in W^{(\gamma)} \text{and} (g', N', s') \in \Omega'_T\}$. For each $(\gamma) \in \Omega_T'^{(v)}$, let $k^*(\gamma)$ denote the index of a (not necessarily unique) secret key s^* in $W^{(\gamma)}$ such that $(g', N', s'_A) \in \Omega_T$. From hereon we denote the outcome triple $\gamma_s \stackrel{\text{def}}{=} (g, N, s_A)$.

For each $(\gamma) \in \Omega_T'^{(v)}$, define the matrix $P^{(\gamma)}$ of size $W^{(\gamma)} + 1$ rows by $W^{(\gamma)}$ columns, whose j'th column is the conditional probability distribution of the output ratio z given that A's secret key is the j'th key in $W^{(\gamma)}$, i.e.:

$$P_{ij}^{(\gamma)} \stackrel{\text{def}}{=} \Pr[T \cap (z = s^{(\gamma)}(i))|(\gamma_s) = (g', N', s^{(\gamma)}(j))] \text{if } i \leq |W^{(\gamma)}| \tag{28}$$

and $P_{ij}^{(\gamma)} \stackrel{\text{def}}{=} \Pr[T \cap (z \neq s^{(\gamma)}(i) \text{for all} i \in \{1,...,W^{(\gamma)}\}) | (\gamma_s) = (g', N', s^{(\gamma)}(j)))]$ if $i = |W^{(\gamma)}| + 1$. Note that the columns of $P^{(\gamma)}$ do not sum to 1 because we have not included in them the conditional probability that $z = xx$ (the outcomes when the execution is not in T). We observe that by (27), at least one column of $P^{(\gamma)}$ (whose index we denote $k^*(\gamma)$) has a sum satisfying

$$\sigma(\gamma) \stackrel{\text{def}}{=} \sum_{i=1}^{|W^{(\gamma)}|+1} P_{ik^*(\gamma)}^{(\gamma)} \geq \epsilon_3/2. \tag{29}$$

Also, using the WI of $\mathcal{SCF}$, (lemma 3 in Appendix) we can show that every pair of columns of $P^{(\gamma)}$ are close in the statistical distance (1-metric) sense:

$$\sum_{i=1}^{|W^{(\gamma)}|+1} |P_{ik}^{(\gamma)} - P_{il}^{(\gamma)}| \leq B \tag{30}$$

where $B \stackrel{\text{def}}{=} (2l_{\mathsf{A-S}}) \cdot (2/2^{k'})$. The result 30 is obtained as follows. The lemma 3 in the appendix gives the bound $(2/2^{k'})$ on the statistical distance between the conditional probability distributions of *single* signcryptext views $\Pr[(e,y,c)|s_A = s_1]$ and $\Pr[(e,y,c)|s_A = s_2]$, given that A's secret key is s_1 and s_2 respectively, and s_1, s_2 are in the same public key set $W^{(\gamma)}$. In our case, the output ratio z can be written as a function $f(\omega, (sc'_1, ..., sc'_{\mathsf{A-S}}), (sc''_1, ..., sc''_{\mathsf{A-S}}))$ (where $f(.)$ is determined by the attacker algorithm A' and Fact) over the space of all inputs and random outcomes in Fact (with the exception of the signcryption algorithm's random input), which we denote collectively by ω, and the collection of up to $2l_{\mathsf{A-S}}$ distinct signcryptexts $((sc'_1, ..., sc'_{\mathsf{A-S}}$ and $(sc''_1, ..., sc''_{\mathsf{A-S}}))$ returned as a result of the signcryption queries of the two attacker runs, respectively. An application of lemma 1 (see appendix) to $f(.)$ shows that the statistical distance between the conditional distributions $\Pr[z = f(\omega, (sc'_1, ..., sc'_{\mathsf{A-S}}), (sc''_1, ..., sc''_{\mathsf{A-S}}))|s_A = s_1]$ and $\Pr[z = f(\omega, (sc'_1, ..., sc'_{\mathsf{A-S}}), (sc''_1, ..., sc''_{\mathsf{A-S}}))|s_A = s_2]$ (which is the distance required in (30)) is not greater than the s.d. between the conditional distributions of the arguments to $f(.)$, ie $\Pr[\omega, (sc'_1, ..., sc'_{\mathsf{A-S}}), (sc''_1, ..., sc''_{\mathsf{A-S}})|s_A = s_1]$ and $\Pr[\omega, (sc'_1, ..., sc'_{\mathsf{A-S}}), (sc''_1, ..., sc''_{\mathsf{A-S}})|s_A = s_2]$. This s.d. can be bounded by repeated application of the combining lemma (Lemma 2 in Appendix) and the $\mathcal{SCF}$ single signcryptext view WI lemma (Lemma 3) as follows. Starting with the identical conditional distributions $\Pr[\omega|s_A = s_1]$ and $\Pr[\omega|s_A = s_2]$ (since ω is independent of s_A), we define the RVs $x^{(1)} = \omega|_{s_A=s_1}$ and $x^{(2)} = \omega|_{sA=s_2}$, $y^{(1)} = (\omega, sc'_1)|_{s_A=s_1}$ and $y^{(2)} = (\omega, sc'_1|_{sA=s_2}$. Also we define the RVs $m_j^{'(i)}$ and $m_k^{''(i)}$ as the $j'th$ and k'th message queries of A' to S in the first and second attacker runs respectively, the supercript $i \in \{1,2\}$ denoting A's secret key as above. We denote probability distributions using the same notation as in Appendix. We observe that (i) $\Delta(D_{x^{(1)}}, D_{x^{(2)}}) = 0$ and (ii) If $x^{(1)} = x^{(2)}$ then $m_1^{'(1)} = m_1^{'(2)}$ so that lemma 3 applies and gives $\Delta(D_{y^{(1)}|x}, D_{y^{(2)}|x}) \leq 2/2^{k'}$. Then (i) and (ii) and Lemma 2 give $\Delta(D_{y^{(1)}}, D_{y^{(2)}}) \leq 0 + 2/2^{k'}$. Now we continue in this manner, defining $x_i^{(1)} = (\omega, sc'_1, ..., sc'_i)|_{s_A=s_1}$ and $x_i^{(2)} = (\omega, sc'_1, ..., sc'_i)|_{sA=s_2}$, $y_i^{(1)} =$

$(\omega, sc'_1, ..., sc'_{i+1})|_{s_A=s_1}$ and $y_i^{(2)} = (\omega, sc'_1, ..., sc'_{i+1})|_{sA=s_2}$ and using lemma 3 and lemma 2 to get $\Delta(D_{y_i^{(1)}}, D_{y_i^{(2)}}) \leq \Delta(D_{x_i^{(1)}}, D_{x_i^{(2)}}) + 2/2^{k'} \leq i \cdot (2/2^{k'})$, for each $i \in \{1, ..., l_{\mathsf{A-S}}\}$. In the same way we can 'add in' the $l_{\mathsf{A-S}}$ queries of the second attacker run, increasing the s.d. bound by $2/2^{k'}$ for each added query, so we get the final upper bound $2l_{\mathsf{A-S}} \cdot (2/2^{k'})$ on the s.d. between the complete views $(\omega, sc'_1, ..., sc'_{l_{\mathsf{A-S}}}), sc''_1, ..., sc''_{l_{\mathsf{A-S}}})$ of Fact with respect to the two secret keys s_1 and s_2, the result expressed in (30). We now have

$$\Pr[T \cap (z = s_A) \cap ((\gamma_s) \in \Omega_T'^{(v)})] = \tag{31}$$

$$\sum_{(\gamma) \in \Omega_T'^{(v)}} \sum_{k=1}^{|W^{(\gamma)}|} \Pr[T \cap (z = s^{(\gamma)}(k))|(\gamma_s) = (g', N', s^{(\gamma)}(k))]$$

$$\cdot \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k))] \tag{32}$$

$$= \sum_{(\gamma) \in \Omega_T'^{(v)}} \sum_{k=1}^{|W^{(\gamma)}|} P_{kk}^{(\gamma)} \cdot \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k))] \tag{33}$$

and

$$\Pr[T \cap ((\gamma_s) \in \Omega_T'^{(v)})]$$

$$= \sum_{(\gamma) \in \Omega_T'^{(v)}} \sum_{k=1}^{|W^{(\gamma)}|} \Pr[T|(\gamma_s) = (g', N', s^{(\gamma)}(k))] \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k))]$$

$$= \sum_{(\gamma) \in \Omega_T'^{(v)}} \sum_{k=1}^{|W^{(\gamma)}|} \sum_{l=1}^{|W^{(\gamma)}|+1} P_{lk}^{(\gamma)} \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k))] \tag{34}$$

so, subtracting (31) from (34), we obtain the lower bound for the for the probability of the desired event

$$\Pr[T \cap (z \neq s_A)] \geq \Pr[T \cap (z \neq s_A) \cap ((\gamma_s) \in \Omega_T'^{(v)})]$$

$$= \Pr[T \cap ((\gamma_s) \in \Omega_T'^{(v)})] - \Pr[T \cap (z = s_A) \cap ((\gamma_s) \in \Omega_T'^{(v)})]$$

$$= \sum_{(\gamma) \in \Omega_T'^{(v)}} \sum_{k=1}^{|W^{(\gamma)}|} \left(\left(\sum_{l=1}^{|W^{(\gamma)}|+1} P_{lk}^{(\gamma)} \right) - P_{kk}^{(\gamma)} \right) \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k))]$$

$$= \sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k_0))] \left(\left(\sum_{k=1}^{|W^{(\gamma)}|} \sum_{l=1}^{|W^{(\gamma)}|+1} P_{lk}^{(\gamma)} \right) - \sum_{k=1}^{|W^{(\gamma)}|} P_{kk}^{(\gamma)} \right)$$

$$\text{for any } k_0 \in \{1, ..., |W^{(\gamma)}|\} \tag{35}$$

$$\geq \sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k_0))]((|W^{(\gamma)}| \cdot \sigma(\gamma) - (|W^{(\gamma)}| - 1) \cdot)$$

$$- \sigma(\gamma) + (|W^{(\gamma)}| - 1) \cdot B) \tag{36}$$

$$= \sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k_0))] \left((\sigma(\gamma) - 2B) \left(|W^{(\gamma)}| - 1 \right) \right) \tag{37}$$

$$\geq (\epsilon_3/2 - 2B) \left(\sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k_0))] \cdot \left(|W^{(\gamma)}| - 1 \right) \right) \tag{38}$$

$$\geq (\epsilon_3/2 - 2B) \left(\sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k_0))] \cdot \left(|W^{(\gamma)}| - 1 \right) \right) \tag{39}$$

$$\geq (\epsilon_3/2 - 2B) \cdot \left(\sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(g, N, v_A) = (\gamma)] \cdot \frac{\left(|W^{(\gamma)}| - 1 \right)}{|W^{(\gamma)}|} \right) \tag{40}$$

$$\geq (\epsilon_3 - 2B)/4 \cdot \left(\sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(g, N, v_A) = (\gamma)] \right) \tag{41}$$

$$\geq (\epsilon_3 - 2B)\epsilon_3/8. \tag{42}$$

To get (35), we have used the fact that $\mathsf{GenUser}$ chooses s_A uniformly in $\{0, ..., S-1\}$ so that $\Pr[(\gamma_s) = (g', N', s^{(\gamma)}(k))] = (1/S) \cdot \Pr[(g, N) = (g', N')]$ is independent of k. To get (36) and (38), we used the WI bound (30) and the existence (29) of a column (having index $k^*(\gamma)$) of sum $\sigma(\gamma)$ exceeding $\epsilon_3/2$:

$$\sum_{k=1}^{|W^{(\gamma)}|} \sum_{l=1}^{|W^{(\gamma)}|+1} P_{lk}^{(\gamma)} = \sum_{k=1}^{|W^{(\gamma)}|} \sum_{l=1}^{|W^{(\gamma)}|+1} ((P_{lk}^{(\gamma)} - P_{lk^*(\gamma)}^{(\gamma)}) + P_{lk^*(\gamma)}^{(\gamma)})$$
$$\geq (|W^{(\gamma)}| - 1) \cdot (\sigma(\gamma)). \tag{43}$$

Similarly, we also used the WI bound (30)in a weaker sense to conclude that each 'diagonal' element $P_{kk}^{(\gamma)}$ exceeds the element $P_{lk^*(\gamma)}^{(\gamma)})$ on the same row in column $k^*(\gamma)$ by at most B, giving

$$\sum_{k=1}^{|W^{(\gamma)}|} P_{kk}^{(\gamma)} \leq \sigma(\gamma) + (|W^{(\gamma)}| - 1)B. \tag{44}$$

To get (41), we used the fact that $S \geq 2\mathrm{Ord}_{\mathbb{Z}_N^*}(g)$ so $|W^{(\gamma)}| \geq 2$ for all (γ) of non-zero probability of being output by $\mathsf{GenComm}$. To get (42), we used (26):

$$\sum_{(\gamma) \in \Omega_T'^{(v)}} \Pr[(g, N, v_A) = (\gamma)] \geq \Pr[\Omega_T] \geq \epsilon_3/2. \tag{45}$$

Finally, we have the claimed lower bound on the success probability of the factoring experiment **FactorExp**:

$$\Pr[\mathbf{FactorExp} = 1] = \Pr[T \cap L \neq 0] \tag{46}$$

$$\geq \Pr[T \cap (z \neq s_A)] - \Pr[s_B = 0] \quad (47)$$
$$\geq (\epsilon_3 - 2B)\epsilon_3/8 - 1/S_L. \quad (48)$$

This establishes the claimed bound and completes the proof. □

The following is a definition of the factorization intractability property we would like to have for the common parameter generation algorithm GenComm of the scheme $\mathcal{SCF}$.

Definition 4. *The common parameter generation algorithm* GenComm *of the scheme* $\mathcal{SCF}$ *is said to be* factorization-intractable *if, for any polynomial time factoring algorithm* Fact$(.,.,.,.)$*, the factoring success probability* $\Pr[\textbf{FactorExp}(k, \mathsf{GenComm}, \mathsf{Fact}) = 1]$ *(where* **FactorExp** *is the experiment defined in Theorem 1) is a negligible function in* k*.*

Now we can state our unforgeability result for the scheme $\mathcal{SCF}$ as a corollary to Theorem 1.

Corollary 1. *For the scheme* $\mathcal{SCF}$*=(*GenComm,GenUser,S,U*), if the common parameter generation algorithm* GenComm *is factorization-intractable, then* $\mathcal{SCF}$ *is existentially unforgeable under an adaptive chosen message attack with respect to random oracle replacement set* $\{1, 2\}$*.*

Proof. Using the fact that $1/2^{k'}$, $1/S$, $1/2^{|\mathsf{H}_2|}$, and $1/2^{|\mathsf{H}_1^{\mathsf{U}}|}$ are all negligible functions in k by construction, and that $l_{\mathsf{A-S}}$, $l_{\mathsf{A-H}_1}$, and $l_{\mathsf{A-H}_2}$ are all polynomially bounded functions in k when the attacker A has a polynomial time bound T (recall each oracle query by A is counted as one computation step), one concludes from Theorem 1 that the existence of a poly-time forging attacker A for $\mathcal{SCF}$ having non-negiligible success probability, implies the existence of a poly-time factoring algorithm for the output of GenComm, having non-negligible success probability. We therefore have a contradiction with the assumption that GenComm is factorization-intractable, so a poly-time attacker A with non-negligible success probability does not exist. □

6 Conclusions

We presented a new signcryption scheme and proved its unforgeability in the random oracle model with respect to a factorization problem. Open problems related to our scheme are: (i) To prove the confidentiality of our scheme relative to the Diffie-Hellman problem or more preferably, relative to the factorization problem and (ii) To find an efficient distributed common parameter generation algorithm for our scheme which leaks no knowledge about the factors of N to a minority of colluding participants. Finally, a remaining problem is to find a scheme at least as efficient as the proposed one which also satisfies one or both of the following properties: (i) The scheme is based on the standard RSA

modulus factorization problem and (ii) Each user has a personal modulus - i.e. the modulus to be factored is not common to all users.

Acknowledgements. The authors would like to thank the anonymous referees for their helpful comments.

References

1. M. Bellare, A. Boldyreva, and S. Micali. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274, Berlin, 2000. Springer-Verlag.
2. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. In *CRYPTO'97*, volume 1294 of *LNCS*, pages 425–439, Berlin, 1997. Springer-Verlag.
3. S. Cavallar et al. Factorization of a 512-Bit RSA Modulus. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 1–18, Berlin, 2000. Springer-Verlag.
4. T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Tran. Info. Theory*, IT-31(4):469–472, 1985.
5. U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *Proc. 22-nd STOC*, pages 416–426. ACM, 1990.
6. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Berlin, 1987. Springer-Verlag.
7. FIPS 186-2, Digital Signature Standard. *Federal Information Processing Standards Publication 186-2*, 2000. Available from `http://csrc.nist.gov/`.
8. M. Girault. Self-Certified Public Keys. In *EUROCRYPT '91*, volume 547 of *LNCS*, pages 490–497, Berlin, 1992. Springer-Verlag.
9. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptively Chosen Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
10. A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. In *PKC2000*, volume 1751 of *LNCS*, pages 446–465, Berlin, 2000. Springer-Verlag.
11. D. Pointcheval. The Composite Discrete Logarithm and Secure Authentication. In *PKC2000*, volume 1751 of *LNCS*, pages 113–128, Berlin, 2000. Springer-Verlag.
12. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. of Cryptology*, 1999. Available from `http://www.di.ens.fr/~pointche`.
13. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 422–436, Berlin, 1998. Springer-Verlag.
14. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *PKC 2000*, volume 1751 of *LNCS*, pages 147–166, Berlin, 2000. Springer-Verlag.
15. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–251, Berlin, 1990. Springer-Verlag.
16. SEC2. *Recommended Elliptic Curve Domain Parameters*, September 2000. Standards for Efficient Cryptography Group. Available from `http://www.secg.org/`.
17. Y. Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). In *CRYPTO'97*, volume 1294 of *LNCS*, pages 165–179, Berlin, 1997. Springer-Verlag.

7 Appendix

This appendix contains several Lemmas used in the security proof of our proposed scheme in section 5.3. In the following, we denote by $D : \Omega \rightarrow \mathbb{R}$ a probability distribution on probability space (P.S.) Ω. The *support* of distribution D on P.S. Ω is the set $\mathrm{Supp}(D) \stackrel{\mathrm{def}}{=} \{x \in \Omega : D(x) \neq 0\}$. A measure of closeness between two probability distributions D_1 and D_2 on a common probability space Ω is the *statistical distance* (s.d.) $\Delta(D_1, D_2) \stackrel{\mathrm{def}}{=} \sum_{x \in \Omega} |D_1(x) - D_2(x)|$.

Lemma 1. *Let $x^{(1)}$ and $x^{(2)}$ be RVs over common P.S. Ω with probability distributions $D_{x^{(1)}}$ and $D_{x^{(2)}}$, respectively. Let $f : \Omega \rightarrow Y$ be an arbitrary mapping. Define the RVs $y^{(1)} \stackrel{\mathrm{def}}{=} f(x^{(1)})$ and $y^{(2)} \stackrel{\mathrm{def}}{=} f(x^{(2)})$ over P.S. Y with resulting probability distributions $D_{y^{(1)}}$ and $D_{y^{(2)}}$, respectively. Then*

$$\Delta\left(D_{y^{(1)}}, D_{y^{(2)}}\right) \leq \Delta\left(D_{x^{(1)}}, D_{x^{(2)}}\right) \tag{49}$$

Proof. Refer to the full paper. □

Lemma 2. *Let $x^{(1)}$ and $x^{(2)}$ be RVs over common P.S. Ω_x, $y^{(1)}$ and $y^{(2)}$ be RVs over common P.S. Ω_y with probability distributions $D_{x^{(1)}}$, $D_{x^{(2)}}$, $D_{y^{(1)}}$, $D_{y^{(2)}}$, respectively. Suppose that $\Delta(D_{x^{(1)}}, D_{x^{(2)}}) \leq B_x$. Define the set $S \stackrel{\mathrm{def}}{=} \mathrm{Supp}(D_{x^{(1)}}) \cap \mathrm{Supp}(D_{x^{(2)}})$. For each $x \in S$, define the conditional distributions $D_{y^{(1)}|x}$ and $D_{y^{(2)}|x}$ by $D_{y^{(i)}|x}(y) \stackrel{\mathrm{def}}{=} \Pr\left[y^{(i)} = y | x^{(i)} = x\right]$ and suppose that $\Delta(D_{y^{(1)}|x}, D_{y^{(2)}|x}) \leq B_y$ for all $x \in S$. Define the ordered-pair RVs $z^{(1)} \stackrel{\mathrm{def}}{=} (x^{(1)}, y^{(1)})$ and $z^{(2)} \stackrel{\mathrm{def}}{=} (x^{(2)}, y^{(2)})$ with resulting distributions $D_{z^{(1)}}$ and $D_{z^{(2)}}$, respectively. Then*

$$\Delta\left(D_{z^{(1)}}, D_{z^{(2)}}\right) \leq B_x + B_y$$

Proof. Refer to the full paper. □

Lemma 3. ***WI*** *For the signcryption scheme $\mathcal{SCF} = (\mathsf{GenComm}, \mathsf{GenUser}, \mathsf{S}, \mathsf{U})$, define the single signcryptext viewing experiment*

Experiment ViewSCF$(k, (g, N), (s_A, v_A), (s_B, v_B), m, \mathsf{H}_1, \mathsf{H}_2, \mathsf{E})$
$(c, e, y) \leftarrow \mathsf{S}(k, g, N, s_A, v_B, m | \mathsf{H}_1, \mathsf{H}_2, \mathsf{E})$
Return (c, e, y)

Suppose $(g, N, S) \in \mathrm{Supp}(\mathsf{GenComm}(k))$, $(s_B, v_B) \in \mathrm{Supp}(\mathsf{GenUser}(k, g, N, S))$ and that H_1, H_2, E are arbitrary oracles and m is an arbitrary message. Let $s_A^{(1)}$ and $s_A^{(2)}$ denote two secret keys in $\{0, \ldots, S-1\}$ which map to the same public key $v_A = g^{-s_A^{(1)}} = g^{-s_A^{(2)}} \bmod N$. For $i \in \{1, 2\}$, define the RVs

$$z^{(i)} \stackrel{\mathrm{def}}{=} \mathbf{ViewSCF}(k, (g, N), (s_A^{(i)}, v_A), (s_B, v_B), m, \mathsf{H}_1, \mathsf{H}_2, \mathsf{E})$$

with corresponding probability distributions $D_{z^{(i)}}$. Then

$$\Delta\left(D_{z^{(1)}}, D_{z^{(2)}}\right) < 2/2^{k'}.$$

Proof. Write $z^{(i)} = (c^{(i)}, w^{(i)})$, with $w^{(i)} = (e^{(i)}, y^{(i)})$, for $i \in \{1, 2\}$. The result $\Delta\left(D_{w^{(1)}}, D_{w^{(2)}}\right) < 2/2^{k'}$ follows from Pointcheval's Theorem 7 [11], since $e^{(i)} = \mathsf{H}_2(\mathsf{H}_1^{\mathsf{U}}(v_B^{r^{(i)}}), m, bind) = \mathcal{S}(g^{r^{(i)}})$, where $\mathcal{S}(.)$ is a probabilistic 'strategy' function which is independent of i. The lemma then follows using Lemma 2 when we observe that $c^{(i)} = \mathsf{E}(\mathsf{H}_1^{\mathsf{L}}(v_B^{r^{(i)}}), m) = f(g^{r^{(i)}})$ for a probabilistic function $f(.)$ independent of i, and $g^{r_i} = v_A^{e^{(i)}} g^{y^{(i)}}$ is determined by $w^{(i)}$, so $\Delta\left(D_{c^{(1)}|w}, D_{c^{(2)}|w}\right) = 0$ for all $w \in \mathrm{Supp}(D_{w^{(1)}}) \cap \mathrm{Supp}(D_{w^{(2)}})$. □

Lemma 4. ***Splitting Lemma*** *Let (a, b) be a pair of RVs on the product PS $\Omega_a \times \Omega_b$. Suppose S is a subset of $\Omega_a \times \Omega_b$ with $\Pr[(a, b) \in S] \geq \epsilon$. Then there exists a subset Ω of Ω_A with the following properties:*

(i)$\Pr[a \in \Omega | (a, b) \in S] \geq 1/2$

(ii) For all $x \in \Omega$, $\Pr[(a, b) \in S | a = x] \geq \epsilon/2$.

Proof. See Pointcheval and Stern's paper [12]. □

Lemma 5. *Define the experiment*

Experiment FindCollExp(FindColl)
Pick a random function $\mathsf{H} : \{0,1\}^* \rightarrow \{0, ..., 2^{|\mathsf{H}|} - 1\}$
$(x_1, x_2) \leftarrow \mathsf{FindColl}(|\mathsf{H})$
If $\mathsf{H}(x_1) = \mathsf{H}(x_2)$ and $x_1 \neq x_2$ **Return** 1
Else **Return** 0

For any algorithm FindColl(|.) *such that* FindColl(|H) *makes up to l queries to oracle* H*,* $\Pr[\mathbf{FindCollExp}(\mathsf{FindColl}) = 1] \leq ((l+1)(l+2)/2) \cdot (1/2^{|\mathsf{H}|})$.

Proof. Refer to the full paper. □

The following lemma is a slight generalization of the factoring algorithm presented by Pointcheval [11], and can be considered a variant of Miller's factoring algorithm (see [14] - the difference is essentially that Miller's original algorithm picks g of order dividing the given L while for this algorithm g is supplied as an input). An *asymmetric basis* g in $\mathbb{Z}_N^*$, where $N = pq$ is an RSA modulus, satisfies $m_2(\mathrm{Ord}_{\mathbb{Z}_p^*}(g)) \neq m_2(\mathrm{Ord}_{\mathbb{Z}_q^*}(g))$, where $m_2(z)$ denotes the multiplicity of 2 in z. Let $odd(z) = z/2^{m_2(z)}$.

Lemma 6. *There exists an algorithm which, given an RSA modulus N, an asymmetric basis g in $\mathbb{Z}_N^*$ and a non-zero multiple L of odd*$(\mathrm{Ord}_{\mathbb{Z}_N^*}(g))$*, outputs a non-trivial factor of N in time $O(|L| \cdot |N|^2)$.*

Proof. Consider the following algorithm.

Algorithm MillFact(g, L, N)

$x_0 \leftarrow g^{\mathrm{odd}(L)} \bmod N$
$i \leftarrow 0$
Repeat

$x_{i+1} \leftarrow x_i^2 \bmod N$
$i \leftarrow (i+1)$
Until $x_i = 1$
$p \leftarrow \gcd(x_{i-1}+1, N)$
Return p

Let $\alpha_p = m_2(\mathrm{Ord}_{\mathbb{Z}_p^*}(g))$ and $r_p = \mathrm{odd}(\mathrm{Ord}_{\mathbb{Z}_p^*}(g))$, with analogous quantities defined for q, and let $\alpha \stackrel{\mathrm{def}}{=} \max(m_2(\mathrm{Ord}_{\mathbb{Z}_p^*}(g)), m_2(\mathrm{Ord}_{\mathbb{Z}_q^*}(g))$. It is easy to show that $\mathrm{Ord}_{\mathbb{Z}_N^*}(g) = \mathrm{lcm}(\mathrm{Ord}_{\mathbb{Z}_p^*}(g), \mathrm{Ord}_{\mathbb{Z}_p^*}(g))$ so that $\mathrm{Ord}_{\mathbb{Z}_N^*}(g) = 2^\alpha \mathrm{lcm}(r_p, r_q)$. From the assumption on L, we see that $\mathrm{odd}(L)$ is an odd multiple of $\mathrm{lcm}(r_p, r_q)$. Therefore, the exponent of g in computing x_i, namely $2^i \mathrm{odd}(L)$, is a multiple of $\mathrm{Ord}_{\mathbb{Z}_p^*}(g)$ iff $i \geq \alpha_p$ and is a multiple of $\mathrm{Ord}_{\mathbb{Z}_q^*}(g)$ iff $i \geq \alpha_q$. But g is by assumption an asymmetric basis in $\mathbb{Z}_N^*$, so one can assume, without loss of generality, that $\alpha = \alpha_q > \alpha_p$. It follows that for $i = i^* \stackrel{\mathrm{def}}{=} \alpha_q - 1$, we have $x_{i^*} = 1 \bmod p$ and $x_{i^*} \neq 1 \bmod q$ but $x_{i^*+1} = 1 \bmod q$. Therefore x_{i^*} is a non-trivial square-root of 1 in $\mathbb{Z}_N^*$, i.e. $x_{i^*} \neq \pm 1 \bmod N$, and the algorithm returns a non-trivial factor of N (due to the well-known simple result: if $a^2 = b^2 \bmod N$ and $a \neq \pm b \bmod N$ then $1 < \gcd(a \pm b, N) < N$) after α squaring operations mod N (time $O(\alpha \cdot |N|^2)$), a modular exponentiation $g^{\mathrm{odd}(L)} \bmod N$ (time $O((|L| - \alpha)|N|^2)$) and a gcd computation (time $O(|N|^2)$). □