Description of Provably Secure Signcryption Schemes

Joonsang Baek[1] and Yuliang Zheng[2]
[1] School of Network Computing, Monash University, Frankston, VIC 3199, Australia
joonsang.baek@infotech.monash.edu.au
[2] Dept. of Software and Info. Systems, University of North Carolina at Charlotte, NC 28223, USA
yzheng@uncc.edu

Aug. 2002

Abstract

In this report we provide a detailed description of four signcryption schemes all of which have been proven secure. The first two of the schemes are based on discrete logarithm on finite fields and the other two schemes are based on elliptic curves. We also discuss issues on the implementation of signcryption schemes.

1 Introduction

Signcryption proposed by Zheng [7] at Crypto'97 is a public key or asymmetric cryptographic method that provides simultaneously both message confidentiality and unforgeability at a lower computational and communication overhead than doing signature and public key encryption separately. Recent progress in the security analysis of signcryption indicates that the specific instantiations of signcryption demonstrated in [7] are indeed secure in a very strong sense. More specifically, it has been proven in [4, 5] that these schemes are secure against adaptive chosen ciphertext attacks and existentially unforgeable against adaptive chosen message attacks, both in the random oracle model, relative to Gap Diffie-Hellman and Strong Discrete Logarithm problems respectively.

It should be emphasized that the signcryption schemes could be proven secure *without any significant changes* of the schemes. However to simplify analysis, [4, 5] modified the original schemes slightly by introducing an extra one-way hashing into the signcryption and unsigncryption operations.

The aim of this report is to provide a detailed description of the signcryption schemes which are provably secure. The reader is referred to [**?**, 5] for detailed explanations on the security proofs of signcryption.

2 Preliminaries

2.1 Notations

In this section we define general notations used in this report.

- $A(\cdot)$ – Algorithm
  - When input is explicitly appeared instead of '$\cdot$' , say 'a', $A(a)$ represents the output value of $A(\cdot)$ on input a.
  - The term 'algorithm' is sometimes referenced as 'function', e.g., 'hash function'.
- $|\cdot|$ – Bit length of an element or output of an algorithm
- $\mathrm{mod} N$ – Modular reduction under modulus N
- $Ord_N(g)$ – Order of an element g modulo N, i.e., the smallest integer a satisfying $g^a = 1 \bmod N$
- $[a, \ldots, b]$ – An integer range between a and b inclusive
- k – An integer representing security parameter, e.g., 512, 1024
- S – Sender unless stated
- R – Receiver unless stated

2.2 Hash functions and Symmetric Cipher

Two one-way hash functions denoted by $G(\cdot)$ and $H(\cdot)$ are used in signcryption. These two functions are assumed as random oracles [6].

The bit length of $G(\cdot)$ and $H(\cdot)$ are determined by the security parameter k which will be provided as input to a signcryption scheme. More precisely, if k is provided as bit length of a prime modulus p, i.e., $k = |p|$ to the signcryption scheme then $|H(\cdot)| = |q|$, where q is a (large) prime such that $q|(p-1)$. Also $|G(\cdot)|$, which is generally smaller than k, is the same as the bit length of a key used in the symmetric encryption.

Constructions of the hash functions $G(\cdot)$ and $H(\cdot)$ modelled as random oracles can be quite flexible but we recommend Mask Generation Function (MGF)-style implementations defined in the IEEE P1363 standard. For example, $H(\cdot)$ can be implemented as follows.

$$H(x) = hash(x||0)|| \cdots ||hash(x||l-1),$$

where $l = \lceil |H(\cdot)|/|hash(\cdot)| \rceil$ and $hash(\cdot)$ is a conventional hash function such as SHA-1 [3].

Note that if $|G(\cdot)| < |hash(\cdot)|$ we simply truncate the output of $hash(\cdot)$ to get the size of $|G(\cdot)|$.

A symmetric block-cipher denoted by $(E_\alpha(\cdot), D_\alpha(\cdot))$ is used in signcryption. Choices for the symmetric block-cipher are also flexible. For example, one can choose DES [2] or AES [1].

3 Description of Signcryption Scheme Defined on Multiplicative Group

3.1 Overview

Zheng's signcryption scheme proposed in [7] is defined over a subgroup of the multiplicative group $\mathbb{Z}_p^*$. As mentioned earlier, this original signcryption scheme with a slight modification was proven secure in [4, 5].

Actually two signcryption schemes were proposed in [7]. These two schemes differ only in computing one of the components in a signcryptext.

We denote these signcryption schemes defined over a subgroup of $\mathbb{Z}_p^*$ by $\mathcal{SCR}$-type1 and $\mathcal{SCR}$-type2, both of which consist of the following four algorithms.

- $\mathsf{GC}(\cdot)$ – Common Parameter Generation
- $\mathsf{GK}(\cdot)$ – Key Generation
- $\mathsf{SC}(\cdot)$ – Signcryption of message
- $\mathsf{USC}(\cdot)$ – Unsigncryption of signcryptext

In the forthcoming sections we precisely describe each sub-algorithm of $\mathcal{SCR}$-type1 and $\mathcal{SCR}$-type2.

3.2 Common Parameter Generation Algorithm

Common parameters are generated by the algorithm $\mathsf{GC}(k)$ where k is a security parameter. The output of this algorithm denoted by cp contains two safe primes p, q and an integer g with order q modulo p. The party that runs $\mathsf{GC}(k)$ can vary, depending on applications. For example, Certificate Authority (CA) can generate the common parameter cp and send it to the parties S and R. However, if S and R themselves have agreed upon the common parameter generation then one of them can generate cp and send it to the other party. A summary of $\mathsf{GC}(k)$ follows.

```
Common Parameter Generation Algorithm $\mathsf{GC}(k)$

1. Choose $p$ – a large prime
2. Choose $q$ – a large prime such that $q|(p-1)$
3. Randomly choose $g$ – an integer from $[1, \dots p-1]$ such that $Ord_p(g) = q$
4. Return $cp = (p, q, g)$
```

3.3 Key Generation Algorithm

In signcryption, two parties denoted by a sender (S) and a receiver (R) are involved. The sender S and the receiver R generate their own secret/public key pairs denoted by (x_S, y_S) and (x_R, y_R) respectively, using the key generation algorithm described below. Note that the security parameter k and the common parameter cp are provided as input to this algorithm.

```
Key Generation Algorithm $\mathsf{GK}(k, cp)$:

1. Pick $x_S$ uniformly at random from $[1, \dots, q-1]$
2. $y_S = g^{x_S} \bmod p$
3. Return $(x_S, y_S)$
```

In the same way, (x_R, y_R) is generated.

3.4 Signcryption Algorithm

The signcryption algorithm $\mathsf{SC}(\cdot)$ is run by the sender S. The common parameter *cp*, the sender S's secret key x_S, and the receiver R's public key y_R and *bind_info* containing the sender and receiver's public keys (y_S, y_R) are provided as input to this algorithm. We remark that including *bind_info* in the input to the signcryption algorithm was first suggested in [8] and it was shown in [4, 5] that *bind_info* is necessary for the signcryption to be proven secure. As pointed out in [8], *bind_info* could contain strings that uniquely identify the sender S and the receiver R or hash values of the public key of each party. However we assume in this report that *bind_info* contains the concatenation of y_S with y_R. A detailed description of $\mathsf{SC}(cp, m, x_S, y_R, \mathit{bind\_info})$ is as follows.

```
Signcryption Algorithm $\mathsf{SC}(cp, m, x_S, y_R, \mathit{bind\_info})$

1. Pick $x$ uniformly at random from $[1, \ldots, q-1]$
2. $w = y_R^x \bmod p$
3. $K = G(w)$
4. $r = H(m, \mathit{bind\_info}, w)$ where $\mathit{bind\_info} = (y_S, y_R)$
5. $s = x/(r + x_S) \bmod q$ if 'type1' is used, or
   $s = x/(1 + x_S \cdot r) \bmod q$ if 'type2' is used
6. $c = E_K(m)$
7. Return $C = (c, r, s)$
```

3.5 Unsigncryption Algorithm

Now we describe the unsigncryption operation of the signcryptext $C = (c, r, s)$ by the receiver R. Note that the common parameter *cp*, the receiver R's secret key x_S, and the sender S's public key y_S and *bind_info* containing the sender and receiver's public keys (y_S, y_R) are provided as input to the unsigncryption algorithm $\mathsf{USC}(\cdot)$.

```
Unsigncryption Algorithm $\mathsf{USC}(cp, C, x_R, y_S, bind\_info)$

1. Parse $C$ as $(c, r, s)$
2. $w = (y_S \cdot g^r)^{s \cdot x_R} \bmod p$ if 'type1' is used, or
   $w = (g \cdot y_S^r)^{s \cdot x_R} \bmod p$ if 'type2' is used
3. $K = G(w)$
4. $m = D_K(c)$
5. If $r \notin [0, \ldots, q-1]$ or $s \notin [1, \ldots, q-1]$ then return '$Rej$ (reject)'
6. If $r = H(m, bind\_info, w)$ then return $m$. Else output '$Rej$'
```

4 Description of Signcryption Scheme Defined on Elliptic Curves over Finite Fields

4.1 Overview

The security proofs of the signcryption schemes provided in [4, 5] are not only applicable to a subgroup of the multiplicative group $\mathbb{Z}_p^*$ as described in Section 3 but also that of the elliptic curve group over finite fields. Similarly to the case of a multiplicative group, we denote two signcryption schemes defined over elliptic curve groups by $\mathcal{ECSCR}$-type1 and $\mathcal{ECSCR}$-type2, both of which consist of the following four algorithms.

- $\mathsf{GC}_{\mathsf{EC}}(\cdot)$ – Common Parameter Generation
- $\mathsf{GK}_{\mathsf{EC}}(\cdot)$ – Key Generation
- $\mathsf{SC}_{\mathsf{EC}}(\cdot)$ – Signcryption of message
- $\mathsf{USC}_{\mathsf{EC}}(\cdot)$ – Unsigncryption of signcryptext

In the forthcoming sections we precisely describe each sub-algorithm of $\mathcal{ECSCR}$.

4.2 Common Parameter Generation Algorithm

Common parameters are generated by the algorithm $\mathsf{GC}_{\mathsf{EC}}(k)$ where k is security parameter. The output of this algorithm is denoted by cp and this contains; an elliptic curve EC over the finite field of p^m denoted by $GF(p^m)$, where p is a prime and m is a positive integer; a prime q whose size is approximately of $|p^m|$; a point P with order q randomly chosen from the points on EC. Note in the choice of $GF(p^m)$ that p and m satisfy either $p \geq 2^k$ and $m = 1$ or $p = 2$ and $m \geq k$. A summary of $\mathsf{GC}_{\mathsf{EC}}(k)$ follows.

```
Common Parameter Generation Algorithm $\mathsf{GC}_{\mathsf{EC}}(k)$

1. Choose $EC$ – an elliptic curve over the finite field $GF(p^m)$, where either
   $p \geq 2^k$ and $m = 1$ or $p = 2$ and $m \geq k$
2. Choose $q$ – a large prime with $|q| \approx |p^m|$
3. Randomly choose $P$ – a point with order $q$ randomly chosen from the
   points on $EC$
4. Return $cp = (EC, q, P)$
```

4.3 Key Generation Algorithm

The sender S and the receiver R generate their own secret/public key pairs denoted by (x_S, P_S) and (x_R, P_R) respectively, using the key generation algorithm described below. Note that the security parameter k and the common parameter *cp* are provided as input to this algorithm.

```
Key Generation Algorithm $\mathsf{GK}_{\mathsf{EC}}(k, cp)$:

1. Pick $x_S$ uniformly at random from $[1, \ldots, q-1]$
2. $P_S = x_S P$
3. Return $(x_S, P_S)$
```

In the same way, (x_R, P_R) is generated.

4.4 Signcryption Algorithm

The common parameter *cp*, the sender S's secret key x_S, and the receiver R's public key P_R and *bind_info* containing the sender and receiver's public keys (P_S, P_R) are provided as input to the signcryption algorithm $\mathsf{SC}_{\mathsf{EC}}(\cdot)$. A detailed description of $\mathsf{SC}(cp, m, x_S, y_R, bind\_info)$ is as follows.

```
Signcryption Algorithm $\mathsf{SC}_{\mathsf{EC}}(cp, m, x_S, P_R, \mathit{bind\_info})$

1. Pick $x$ uniformly at random from $[1, \ldots, q-1]$
2. $W = xP_R$
3. $K = G(W)$
4. $r = H(m, \mathit{bind\_info}, W)$ where $\mathit{bind\_info} = (P_S, P_R)$
5. $s = x/(r + x_S) \bmod q$ if 'type1' is used, or
   $s = x/(1 + x_S \cdot r) \bmod q$ if 'type2' is used
6. $c = E_K(m)$
7. Return $C = (c, r, s)$
```

4.5 Unsigncryption Algorithm

The common parameter *cp*, the receiver R's secret key x_S, and the sender S's public key y_S and *bind_info* containing the sender and receiver's public keys (P_S, P_R) are provided as input to the unsigncryption algorithm $\mathsf{USC}_{\mathsf{EC}}(\cdot)$.

```
Unsigncryption Algorithm $\mathsf{USC}_{\mathsf{EC}}(cp, C, x_R, P_S, \mathit{bind\_info})$

1. Parse $C$ as $(c, r, s)$
2. $u = sx_R \bmod q$
3. $W = uP_S + urP$ if 'type1' is used, or
   $W = uP + urP_S$ if 'type2' is used
4. $K = G(W)$
5. $m = D_K(c)$
6. If $r \notin [0, \ldots, q-1]$ or $s \notin [1, \ldots, q-1]$ then return '*Rej* (reject)'
7. If $r = H(m, \mathit{bind\_info}, W)$ then return $m$ Else output '*Rej*'
```

References

[1] *Advanced Encryption Standard*, Federal Information Processing Standard Publication 197, 2001.

[2] *Data Encryption Standard (DES)*, Federal Information Processing Standard Publication 46, 1976.

[3] *Secure Hash Standard*, Federal Information Processing Standard Publication 180-1, 1995.

[4] J. Baek, R. Steinfeld and Y. Zheng: *Formal Proofs for the Security of Signcryption*, Proceedings of Public Key Cryptography 2002 (PKC 2002), Vol. 2274 of LNCS, Springer-Verlag 2002, pages 80–98.

[5] J. Baek, R. Steinfeld and Y. Zheng: *Formal Proofs for the Security of Signcryption*, A full version, Submitted to Jornal of Cryptology. A draft is available upon request to the authors.

[6] M. Bellare and P. Rogaway: *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, Proceedings of First ACM Conference on Computer and Communications Security 1993, pages 62–73.

[7] Y. Zheng: *Digital Signcryption or How to Achieve Cost (Signature & Encryption) $\ll$ Cost (Signature) + Cost (Encryption)*, Advances in Cryptology - Proceedings CRYPTO '97, Vol. 1294 of LNCS, Springer-Verlag 1997, pages 165–179.

[8] Y. Zheng: *Digital Signcryption or How to Achieve Cost (Signature & Encryption) $\ll$ Cost (Signature) + Cost (Encryption)*, full version, available at `http://www.pscit.monash.edu.au/ yuliang/pubs/`.