Improving Signcryption Scheme

Jin-Woo Chung, Hong-Yeop Song

Coding & Information Theory Laboratory
Dept. of Electrical and Electronic Engineering, Yonsei University

Contents

- Introduction
- Defects of Signcryption
- Possibility of Improving Signcryption
- Analysis of Signcryption
- Improving Signcryption
- Conclusion

Introduction

- Signcryption ?
  - **Sign**ature + En**cryption**
- Signcryption is based on shortened DSS(Digital Signature Standard)
- Parameters are same as ElGamal-type signature-then-encryption.
- Signcryption can save computational cost as well as communication overhead compared to conventional signature-then-encryption scheme.

Signcryption - Advantage

Various schemes	Computational cost	Communication overhead (in bits)
signature-then-encryption based on RSA	EXP=2, HASH=1, ENC=1 (EXP=2, HASH=1, DEC=1)	$\|n_a\| + \|n_b\|$
signature-then-encryption based on "DSS + ElGamal encryption"	EXP=3, MUL=1, DIV=1 ADD=1, HASH=1, ENC=1 (EXP=2.17, MUL=1, DIV=2 ADD=0, HASH=1, DEC=1)	$2\|q\| + \|p\|$
signature-then-encryption based on "Schnorr signature + ElGamal encryption"	EXP=3, MUL=1, DIV=0 ADD=1, HASH=1, ENC=1 (EXP=2.17, MUL=1, DIV=0 ADD=0, HASH=1, DEC=1)	$\|hash(\cdot)\| + \|q\| + \|p\|$
signcryption SCS1	EXP=1, MUL=0, DIV=1 ADD=1, HASH=2, ENC=1 (EXP=1.17, MUL=2, DIV=0 ADD=0, HASH=2, DEC=1)	$\|KH.(\cdot)\| + \|q\|$
signcryption SCS2	EXP=1, MUL=1, DIV=1 ADD=1, HASH=2, ENC=1 (EXP=1.17, MUL=2, DIV=0 ADD=0, HASH=2, DEC=1)	$\|KH.(\cdot)\| + \|q\|$

Signcryption - Algorithm

Parameters

- Private Key : x_a, x_b
- Public Key : y_a, y_b
- Public Parameter : p, q, g

$(C \| s \| r)$

Signcryption

$1 \le x \le q-1$ ← **Select *x* arbitrarily**

$k = hash(y_b{}^{x} \pmod p)$

Split *k* into k_1 and k_2

$r = KH_{k_2}(m)$

signature

$s \equiv x/(r + x_a) \pmod q$ → **SCS1**

$s \equiv x/(1 + x_a \cdot r) \pmod q$ → **SCS2**

encryption ← $C = E_{k_1}(m)$

Unsigncryption

SCS1 ← $k = hash((y_a \cdot g^r)^{s \cdot x_b} \pmod p)$

SCS2 ← $k = hash((y_a{}^{r} \cdot g)^{s \cdot x_b} \pmod p)$

Split *k* into k_1 and k_2

$D_k(C) = m$ → **decryption**

$KH_{k_2}(m)$

Accept *m* as valid if

$KH_{k_2}(m) = r$

signature verification

Defects of Signcryption

(1) There exists possibility of *s* being divided by zero

- System error.
- Recalculation increases computational cost.

(2) Signcryption needs division algorithm

- Among addition, subtraction, multiplication and division algorithm, the division algorithm is the most difficult one.

Defects of Signcryption – Defect (1)

Signcryption

$1 \le x \le q-1$ ← **Select *x* arbitrarily**

$k = hash(y_b^{\,x} (\mathrm{mod} p))$

Split *k* into k_1 and k_2

$r = KH_{k_2}(m)$

$s \equiv x/(r + x_a)(\mathrm{mod} q)$ ➡ $r + x_a \equiv 0(\mathrm{mod}\, q)$

$s \equiv x/(1 + x_a \cdot r)(\mathrm{mod} q)$ ➡ $1 + r \cdot x_a \equiv 0(\mathrm{mod}\, q)$

$C = E_{k_1}(m)$

➡ **Divided by zero System Error !!**

Recalculation is needed (which includes modulo exponentiation!!)

DSS

$1 \le x \le q-1$ ← **Select *x* arbitrarily**

$r \equiv (g^x (\mathrm{mod}\, p))(\mathrm{mod}\, q)$

$s \equiv (hash(m) + x_a \cdot r)/x(\mathrm{mod}\, q)$

Divided by zero never occur.

➡ **Although signcryption is based on shortened DSS, DSS does not suffer from defect (1)**

Y.Zheng thought changing *s* slightly different from DSS will not pose any problem. But it does !!

Defects of Signcryption – Defect (2)

Signcryption

$1 \le x \le q-1$ ← **Select *x* arbitrarily**

$k = hash(y_b{}^x \pmod p)$

Split *k* into k_1 and k_2

$r = KH_{k_2}(m)$

$s \equiv x/(r + x_a) \pmod q$

$s \equiv x/(1 + x_a \cdot r) \pmod q$

$C = E_{k_1}(m)$

Division is needed in the Calculation of *s*.

The inclusion of division algorithm in the smart card or mobile terminal increases the size as well as processing time.

Therefore, if division can be avoided, it is desirable not to use division algorithm.

Possibility of Improving Signcryption

- Could there be ways to overcome defects ?
  - Defects lies in the calculation of *s*
  - Some known signature schemes do not suffer from defects.
  - If variants of signcryption can be found by generalizing it, there might be some variants that can overcome defects.
- Analyzing signcryption more closely will help eliminating undesirable part of signcryption and improving it.

Analysis of Signcryption

Signcryption

Select *x* arbitrarily

$k = hash(y_b{}^x \pmod p)$ → Must we use one-way hash function?

Split *k* into k_1 and k_2 → Is splitting *k* the only way to obtain k_1 and k_2 ?

$r = KH_{k_2}(m)$

$s \equiv x/(r + x_a) \pmod q$

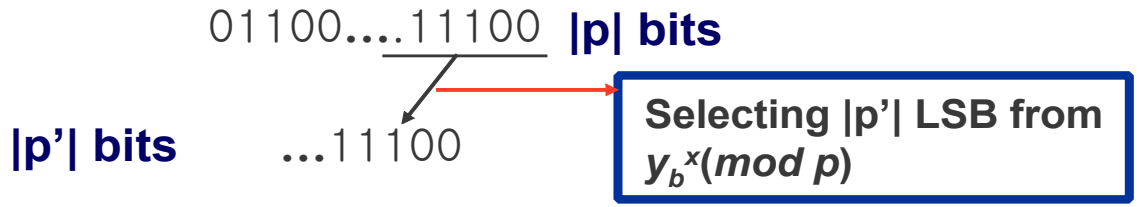
$s \equiv x/(1 + x_a \cdot r) \pmod q$

} Is it possible to generalize the calculation of *s* ?

$C = E_{k_1}(m)$

Analysis of Signcryption – Calculation of k, k_1, k_2

- Arbitrary function can be used instead of hash function in obtaining k.
  - Example) Define function $h : Z_p \rightarrow Z_{p'}$ as selecting |p'| bits from $y_b^x (mod\ p)$



- One of the simple ways of obtaining k_1, k_2 from k is choosing $k=k_1$ and obtain k_2 from k_1.

Analysis of Signcryption – Generalizing signature equation

- Is it possible to generalize the calculation of *s* ?
  - Yes.
  - How ?
    - In ElGamal-type signature, calculation of *s* (which is called signature equation) can be generalized.
    - Signcryption is based on shortened DSS. Since shortened DSS is ElGamal-type signature, there must be some ways to generalize the calculation of *s* (signature equation).

Analysis of Signcryption – Calculation of *s*

- After close consideration of generalization of signature equation, some variants that do not suffer from defect (1) and (2) are found.
- One of variants that do not suffer from defects calculates *s* as follows.

$$s \equiv x - x_a \cdot r \pmod{q}$$

Improving Signcryption – Proposed Scheme

Parameters

- Same as original signcryption scheme.
- Function $h : Z_p \rightarrow Z_{p'}$ is selecting |p'| LSB from $y_b^x (mod\ p)$.

Signcryption $\xrightarrow{(C\|s\|r)}$ **Unsigncryption**

$1 \le x \le q-1$ ← **Select *x* arbitrarily**

$$k = h(y_b^{\,x} (\mathrm{mod}\ p))$$

$k = k_1$ and obtain k_2 from k_1

signature

$$r = KH_{k_2}(m)$$
$$s \equiv x - x_a \cdot r (\mathrm{mod}\ q)$$

encryption ← $C = E_{k_1}(m)$

$$k = h(y_a^{\,r \cdot x_b} \cdot g^{s \cdot x_b} (\mathrm{mod}\ p))$$

Obtain k_1 and k_2

$D_k(C) = m$ → **decryption**

$KH_{k_2}(m)$

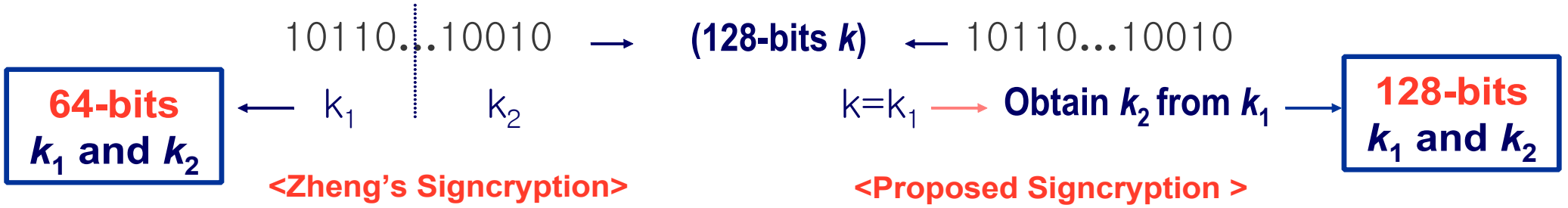
Accept *m* as valid if

$KH_{k_2}(m) = r$

signature verification

Improving Signcryption
- Advantage of Proposed Scheme

1. Computational cost for initial hashing is eliminated.
2. The binary length of k_1 and k_2 is doubled.



3. Signcryption scheme has become flexible.
   - Flexibility enables signcryption to change depending on various situations encountered.

Improving Signcryption – SCS vs Proposed Scheme

- Computational Cost

	EXP	MUL	DIV	ADD/SUB
SCS1	1.17	2	1	1
SCS2	1.17	3	1	1
Proposed	1.17	3	0	1

  - Since *k* can be precalculated, modulo exponentiation in signcryption part is not considered as cost.
  - Division is replaced by multiplication.

Conclusion

- There are many variants of signcryption some of which are more efficient than original signcryption.
- Through generalization of signcryption, several ways to overcome some defects and improve signcryption are found.
- Signcryption scheme has become flexible through analysis. However, it is desirable to have a single standard for the commercial use of signcryption.