A Design for Modular Exponentiation Coprocessor in Mobile Telecommunication Terminals

Takehiko Kato, Satoru Ito, Jun Anzai, and Natsume Matsuzaki

Advanced Mobile Telecommunications Security Technology Research Laboratories Co., Ltd.
BENEX S3 Building 12F, 3-20-8 Shinyokohama, Kohoku-ku, Yokohama, 222-0033 Japan
{tkato,anzai,matuzaki}@amsl.co.jp

Abstract. Following requirements are necessary when implementing public key cryptography in a mobile telecommunication terminal. (1) simultaneous high-speed double modular exponentiation calculation, (2) small size and low power consumption, (3) resistance to side channel attacks. We have developed a coprocessor that provides these requirements. In this coprocessor, right-to-left binary exponentiation algorithm was extended for double modular exponentiations by designing new circuit configuration and new schedule control methods. We specified the desired power consumption of the circuit at the initial design stage. Our proposed method resists side channel attacks that extract secret exponent by analyzing the target's power consumption and calculation time.

1 Introduction

The use of public key cryptography in mobile telecommunication is on the increase. Small size, lightweight and low power consumption are necessary for mobile telecommunication terminals. These devices, because they are small, are easily lost or stolen. They have a risk to be disassembled or analyzed by the third party.

Public key cryptography requires large-scale calculations, using modular exponentiation factors of up to 1024 bits. The low powered MPU used in a typical mobile telecommunication terminal takes a long time to perform these calculations. It can take several seconds to perform a modular exponentiation in software.

There are many cases when double or more modular exponentiations are required in the verification of signature based on discrete log such as DSA [1] or Nyberg-Rueppel signature [2], Cramer-Shoup scheme [3] and Anzai-Matsuzaki- Matsumoto scheme [4][5]. For other examples, RSA use a modular exponentiation, but more modular exponentiations are required to check the certificate of CA.

Recently, there have been examples of side channel attacks, which use information leaked during cryptographic processing. Circuits that are resistant to such attacks are needed. Side channel attacks include power analysis attacks, timing attacks and electromagnetic emission attacks. There are many studies of each of these symmetric cryptographs and public key cryptographs, some of which we will describe next.

Paul Kocher et al. tested timing attacks on Diffie-Hellman, RSA and DSA in [6]. They discovered that by carefully measuring the time required to perform symmetric key operation, attackers could find fixed Diffie-Hellman exponents, could find factor

Ç.K. Koç and C. Paar (Eds.): CHES 2000, LNCS 1965, pp. 216 - 228, 2000.

RSA keys, and broke other cryptography. Messerges et al. examined power analysis attack on the modular exponentiation of public key cryptography in [7]. Goubin et al. studied power analysis attacks on RSA and described countermeasures in [8]. Handschuh et al. tested probing attacks using a monitor oracle in [9]. As we have said, a lot of research is being done on side channel attack method, and coprocessor performing encryption algorithms must be resistant to these types of attacks. To achieve this goal, calculation time should be kept constant and current variation should vary as little as possible.

Therefore, we will develop a coprocessor that fulfils the following requirements:

- simultaneous high-speed double modular exponentiations,
- small size and low power consumption,
- resistance to side channel attacks.

After clearing problems of conventional circuits by basic investigations, we consider countermeasures. However, these countermeasures cannot satisfy our requirements. We propose new method in section 4.

2 Basic Investigations

As shown below, the modular exponentiation calculation T=A^B mod C is performed using the square-and-multiply algorithm. Here, A is base, B is exponent and C is modulus. The left-to-right circuit (LRC) is based upon the left-to-right binary exponentiation algorithm [11] and the right-to-left circuit (RLC) is based upon the right-to-left binary exponentiation algorithm [11]. The RLC process the modular square and modular multiply in parallel.

Now we will compare RLC and LRC in terms of the three requirements mentioned above. Here a "loop" means one modular square calculation or one modular multiply calculation. In LRC, when the B is "0", only a modular square is performed and it loops once. When the B is "1", both a modular square and a modular multiply are performed and it loops twice. On the other hand, in RLC, whether the B is "0" or "1" there is only one loop because of parallel processing.

2.1 Calculation Time, Power Consumption, and Number of Gates

The power consumption of the circuit can be estimated using simulation data available at the circuit design stage. Using requirements in Table.1, we estimated the power consumption when LRC and RLC were installed in an ASIC (Fujitsu CE61).

Table 1. Requirement for Power Consumption Analysis and Simulation

Analysis tool	PROVERD/PWR (Fujitsu LSI technology)
Simulation	Verilog XL
Clock frequency	20MHz (50ns)
Measurement interval	500 ns (per every 10 clocks)

We estimated the current consumption of LRC and RLC using 8 bits B when A and C of 1024 bits each. The results are shown in Table.2. In this paper, the current consumption is also called the power consumption. Current consumption [microsec*mA] equal average current [mA] multiplied by calculation time [microsec]. Number of gates is 28326 for LRC and 37049 for RLC each.

Table 2. Comparison between LRC and RLC on ASIC

	LRC			RLC		
B	Calculation time [microsec]	Average current [mA]	Current Consumption [microsec*mA]	Calculation time [microsec]	Average current [mA]	Current Consumption [microsec*mA]
1000 0000	2240	23.7	52994	2360	31.9	75336
1010 1010	3160	23.7	74927	2370	34.7	82147
1111 0000	3160	23.7	74977	2370	34.6	82019
1111 1111	4400	23.8	104553	2390	38.1	91032

As seen above, the current consumption of LRC and RLC is almost same. The calculation time of RLC is shorter than that of LRC. The number of gates required for of RLC is larger than that of LRC.

2.2 Resistance to Power Analysis Attacks and Timing Attacks

2.2.1 Current Waveform of RLC

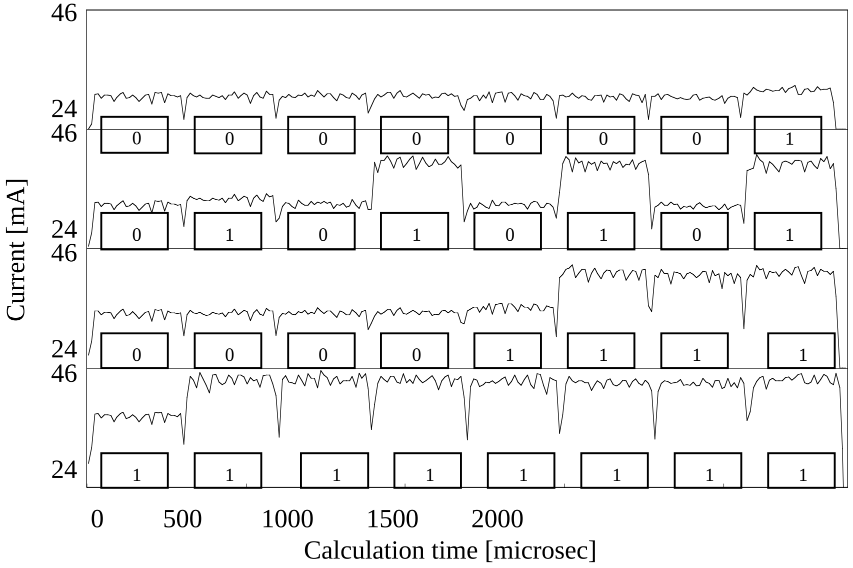

Fig. 1. Current Waveform of RLC

The current waveform of RLC was measured using a base and modulus of 1024 bits each and exponents of 8 bits. From Fig.1, we can readily see the difference in current variation when the exponent is "1" and when it is "0". The variation is high at “1” and low at “0”. By monitoring these fluctuations, we can easily determine the value of exponent. In public key cryptography systems such as RSA, ElGamal, etc., it is critical to keep the exponents secret. If RLC is used, we must provide a way to prevent power analysis attacks. In RLC, the calculation time is constant regardless of the number of "1"s in the exponent, making it resistant to timing attacks.

2.2.2 Current Waveform of LRC

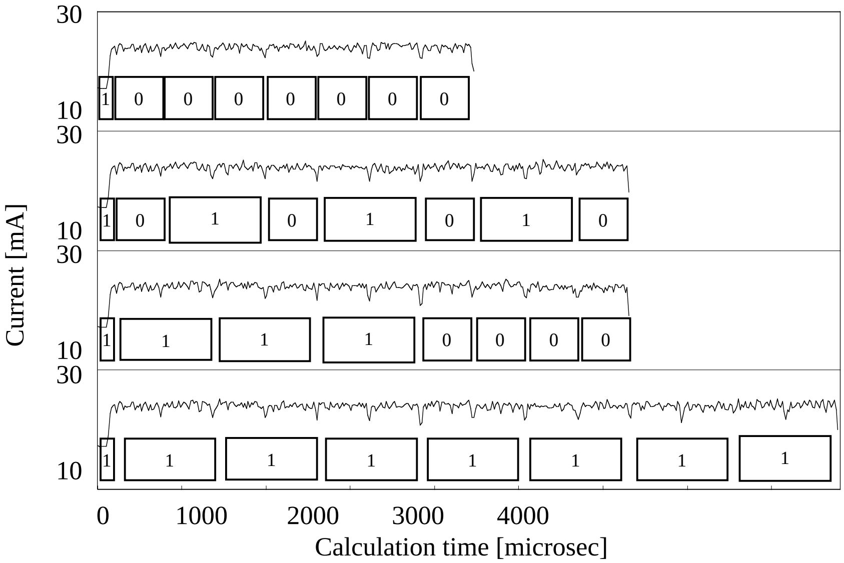

Fig. 2. Current Waveform of LRC

From Fig.2, we can see current variations corresponding to exponents are smaller. It is more difficult to determine the exponent value by monitoring the current variation. However, the calculation time variation can be more easily observed, being proportional to the Hamming weight. This means that although LRC is more resistant to power analysis attacks, it is more vulnerable to timing attacks.

In practice, signal leakage is minute, and is usually masked by noise. Integrated-and dump filters or other technologies are in use [7].

2.3 Overall Comparison between RLC and LRC

Now we will compare RLC and LRC based on the above discussion.

Table 3. Overall Comparison between RLC and LRC

	calculation time	number of gates	power consumption	timing attacks	power analysis attacks
RLC	allowed	not allowed	fairly good	difficult	possible
LRC	not allowed	allowed	fairly good	possible	difficult

RLC has the advantages of calculation time and resistance to timing attacks. On the other hand, LRC has the advantages of fewer gates and resistance to power analysis attacks.

Semiconductor manufacturing technology has great progress in miniaturization, lessening the impact of gate costs. We see calculation time as a more important factor than the number of gates. Our preference, therefore, is RLC and this is what we will subsequently discuss.

3 Countermeasures

We decided to adopt RLC because of its reduced calculation time, in spite of the risk that the modular exponent might be decoded from current waveform analysis. We will perform double modular exponentiations by running two RLC in parallel. This configuration will be called D-RLC. We also considered dual LRC (D-LRC), but these double both the number of gates required and the power consumption.

Multiple modular exponentiation methods were considered in [10], but these systems required a lot of memory, and were considered unsuitable for mobile telecommunication terminals. We studied the faster system of simultaneous double modular exponentiations. Studies are being done on efficient multiple modular exponentiation calculations in [11]. However, most of them need large memory-intensive tables making them unsuitable for mobile telecommunication terminals. For that reason, we didn't take them into consideration.

We considered countermeasure against both power analysis attacks and timing attacks. We used a dummy calculation (DC) to forcibly the RLC to always perform both a modular square circuit and a modular multiply. This DC emulates the idle circuit using previously calculated data for instance, and every circuit is constantly operated even if the exponent is "0".

Using DC, the modular exponentiation calculations in both RLC and LRC show the same current waveforms and current consumption as that of all exponents in “1”. By adjusting the calculation time to give double time for "0" bits, LRC can also be made resistant to timing attacks.

The method that varies calculation time using a blind signature is proposed in [6]. This method is effective for power analysis attacks. Goubin et al. divided plain text into multiple parts and calculated each, and then combined the results. This method is effective against power analysis attacks because it alters the pattern of electromagnetic emission. These last two methods may increase calculation time, number of gates or required MPU processing power.

We can see that for a circuit to be resistant to power analysis attacks and timing attacks, it must operate with constant current variation and calculation time regardless of input values. But this may result in excessive current consumption and calculation time which is a problem in practical use. In the next section, we will discuss a way to make a practical system with sufficient resistance to power analysis attacks and timing attacks.

Therefore, a different approach is necessary. Next, we will show our proposed method.

4 Our Proposed Method (OPM)

Our proposed method consists of a new circuit configuration and a new schedule control method.

4.1 New Circuit Configuration

Using DC, the results were the slowest calculation time or the highest current consumption. We realized that in many cases double modular exponentiation calculations performed for public key cryptography. Modular squares are always performed, but modular multiplies are performed only when the exponent is "1", never when the exponent is "0". This means that shared modular multiply units were a possibility. As shown in Fig.3, we first used two separate RLC. Then we combined the two separate modular multiply units into one for shared use. This results in fewer gates.

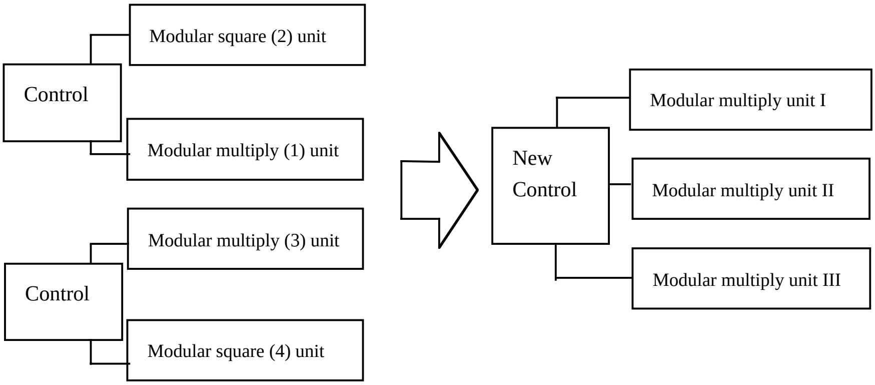

Fig. 3. New Circuit Configuration

The modular multiply unit consists of two 16 bits multipliers using the method that modular calculation per partial multiply are performed. In the new circuit configuration, the shared modular multiply unit II cannot be used for double modular exponentiation calculations when both exponents are equal to "1". In this case, one of the calculations may have to be delayed until the modular multiply unit becomes idle.

4.2 New Schedule Control Method

Here we propose a new method of control scheduling that avoids the delay problem mentioned above.

Double modular exponentiations are divided into two modular squares and two modular multiplies for each i-th bit of exponent B. These four instructions enter the three modular multiply units (I, II and III) in order. But some exception handling is necessary. In the control part, the instructions correspond to exponents are stored in a register FIFO. During the calculation phase, the register FIFO is monitored, and the instructions are executed. Fig.4 shows the process flow.

The control and calculation parts are performed in parallel. In the control part, four instructions ($(1)_i$-$(4)_i$) correspond to exponent i-th bit of B entered four control register (FIFO(0)-FIFO(3)). In the calculation part, three modular multiply units are performed. The FIFO with the under-bar contains the most significant bit calculation, and the asterisk means either calculation.

Two modular squares and two modular multiplies are performed in three modular multiply units for each i-th bit of B in Fig.4. The expression $(1)_i$ or $(3)_i$ shows a modular multiply and the $(2)_i$ or $(4)_i$ shows a modular square in Fig.3. The one bit calculation is carried out in one or two loops. One loop consists of one operation of the three modular multiply units. The shadow part corresponds to the input exponent. The oblique line is uncalculated part. The system disallows:

- a modular square or modular multiply of a different exponent in same loop (ex.$(2)_i$ and $(2)_{i+1}$ in the same loop),
- a modular square and modular multiply of different i-th bit of same exponent in same loop (ex.$(4)_i$ and $(3)_{i+1}$ in the same loop)

We call this prohibition law.

Examples of new schedule control method are shown in Fig.5.

Fig.5 shows the how modular multiply units I and III are able to simultaneously operate when both exponents B1 and B2 are "1". In some cases, however, modular multiply unit III is blocked (see aforementioned prohibition law). We considered avoiding this prohibition law by changing the order of the calculation. In right-to-left binary exponentiation algorithm, the 1 bit result is used in the next calculation. In this case, the modular multiply requires the results of both previous modular multiply and modular square. But the modular square needs only the results from the previous modular square. By preprocessing the modular square and storing the results in two 1024 bits buffers, we can solve the problem. Fig.6 shows to avoid part 2 of the prohibition law (i.e.,$(4)_i$ and $(3)_{i+1}$ in the same loop). As we can see in the 2nd part of Fig.5, modular multiply unit III cannot process the 4th loop $(4)_3$. But we can preprocess $(4)_2$ and replace $(3)_2$ in the 2nd loop(see Fig.6). Then we can process $(4)_3$ in the 3rd loop. The uncalculated part of Fig.5 is replaced by preprocessing.

```
//Length establishment of exponents
 b1msb = MSB(B1); b2msb = MSB(B2);
 last = MAX(b1msb, b2msb);
//Control part
for (i=0 ; i <= last ; i++){
    if (B1(i) == 0 && B2(i) == 0){
                    FIFO(0,1) = ((2),  (4)); }
    if (B1(i) == 1 && B2(i) == 0){
            if (i == last){ FIFO(0)    = (_(1)       ); }
            else   { FIFO(0,1,2) = (  (1), (2), (4)); }}
    if (B1(i) == 0 && B2(i) == 1){
            if (i == last){ FIFO(0)    = (   _(3)   ); }
            else       { FIFO(0,1,2) = ((2), (3), (4)); }}
    if (B1(i) == 1 && B2(i) == 1){
            if (i == last){ FIFO(0,1)    = ((1),   _(3)   ); }
            else   { FIFO(0,1,2,3) = ((1), (2), (3), (4)); }}

//Calculation part

while(1){
    if ( FIFO(0,1) == ((2), (4)) || FIFO(0,1) == ((4), (2)) ||
                            FIFO(0,1,2) == ((4), (1), (3)) ||FIFO(1) == (_*) ){
                    Modular_multiply_unit_I(FIFO(0));
                    Modular_multiply_unit_II(FIFO(1));
    }else if ( FIFO(0,1) == ((2), (1)) || FIFO(0,1) == ((4),  (3)) || FIFO(0) == (_*) ){
                    Modular_multiply_unit_I(FIFO(0));
    }else{
                    Modular_multiply_unit_I(FIFO(0));
                    Modular_multiply_unit_II(FIFO(1));
                    Modular_multiply_unit_III(FIFO(2));}
    if ( FIFO(0) == _* || FIFO(1) == _* || FIFO(2) == _* )
                    break; }}
```

Fig. 4. Calculation Process Flow of New Schedule Control Method

Modular multiply unit	Input exponent B1 =	1	0	0	0	0	0	0	0		
	Input exponent B2 =	1	1	1	1	1	1	1	1		
	Start					End					
I	$(2)_1$	$(2)_2$	$(2)_3$	$(2)_4$	$(2)_5$	$(2)_6$	$(2)_7$	$(1)_8$	▨	▨	▨
II	$(3)_1$	$(3)_2$	$(3)_3$	$(3)_4$	$(3)_5$	$(3)_6$	$(3)_7$	$(3)_8$	▨	▨	▨
III	$(4)_1$	$(4)_2$	$(4)_3$	$(4)_4$	$(4)_5$	$(4)_6$	$(4)_7$	▨	▨	▨	▨

Modular multiply unit	Input exponent B1 =	1	0	1	0	1	0	1	0		
	Input exponent B2 =	1	1	1	1	1	1	1	1		
I	$(2)_1$	$(1)_2$	$(4)_2$	$(3)_3$	$(2)_4$	$(2)_5$	$(1)_6$	$(4)_6$	$(3)_7$	$(3)_8$	▨
II	$(3)_1$	$(2)_2$	$(2)_3$	$(4)_3$	$(3)_4$	$(3)_5$	$(2)_6$	$(2)_7$	$(4)_7$	▨	▨
III	$(4)_1$	$(3)_2$	▨	$(1)_4$	$(4)_4$	$(4)_5$	$(3)_6$	▨	$(1)_8$	▨	▨

Modular multiply unit	Input exponent B1 =	1	1	1	1	0	0	0	0		
	Input exponent B2 =	1	1	1	1	1	1	1	1		
I	$(2)_1$	$(2)_2$	$(2)_3$	$(2)_4$	$(1)_5$	$(4)_5$	$(3)_6$	$(2)_7$	$(1)_8$	▨	▨
II	$(3)_1$	$(3)_2$	$(3)_3$	$(3)_4$	$(2)_5$	$(1)_6$	$(4)_6$	$(3)_7$	$(3)_8$	▨	▨
III	$(4)_1$	$(4)_2$	$(4)_3$	$(4)_4$	$(3)_5$	$(2)_6$	$(1)_7$	$(4)_7$	▨	▨	▨

Modular multiply unit	Input exponent B1 =	1	1	1	1	1	1	1	1		
	Input exponent B2 =	1	1	1	1	1	1	1	1		
I	$(1)_1$	$(4)_1$	$(3)_2$	$(2)_3$	$(1)_4$	$(4)_4$	$(3)_5$	$(2)_6$	$(1)_7$	$(4)_7$	$(3)_8$
II	$(2)_1$	$(1)_2$	$(4)_2$	$(3)_3$	$(2)_4$	$(1)_5$	$(4)_5$	$(3)_6$	$(2)_7$	$(1)_8$	▨
III	$(3)_1$	$(2)_2$	$(1)_3$	$(4)_3$	$(3)_4$	$(2)_5$	$(1)_6$	$(4)_6$	$(3)_7$	▨	▨

Fig. 5. Examples of New Schedule Control Method

Modular multiply unit	Input exponent B2 =			1	0	1	0	1	0	1	0
	Input exponent B3 =			1	1	1	1	1	1	1	1
I	$(2)_1$	$(1)_2$	$(3)_2$	$(3)_3$	$(3)_4$	$(3)_5$	$(2)_6$	$(2)_7$	$(1)_8$	▨	▨
II	$(3)_1$	$(2)_2$	$(2)_3$	$(1)_4$	$(4)_4$	$(4)_5$	$(3)_6$	$(3)_7$	$(3)_8$	▨	▨
III	$(4)_1$	$(4)_2$	$(4)_3$	$(2)_4$	$(2)_5$	$(1)_6$	$(4)_6$	$(4)_7$	▨	▨	▨

Fig. 6. Schedule Replacing Example by Preprocessing

5 Evaluation

We evaluated the method that prevent discovery of modular exponents by third party monitoring of the current consumption patterns and calculation time of the circuitry.

OPM is resistant to timing attacks and power analysis attacks because:

- one bit processing of exponent is spread over one or two loops performed by the modular multiply units,
- various kind of exponents are mixed in the same loop,
- if the exponents are reverse (B1, B2 are reversed B2, B1), the current waveform is changed corresponding to the processing.

The number of loop changes not only based on the combination of the i-th "0" and "1", but also the (i-1)-th combination or the (i+1)-th combination. The calculation time required by OPM does not increase proportionally to the Hamming weight as it does in LRC. The larger the combination of the i-th exponents B1 and B2 per loop is, the larger the safety margin will be. In OPM, even in one loop, there are two possibilities to determine that only i-th bit is performed and (i+1)-th bits are performed. For this reason, it is not possible to determine whether the combination of exponents is "0 and 1", "1 and 0", "1 and 1" or "0 and 0". Fig.7 shows the waveform of OPM corresponding to Fig.5. OPM is resistant to power analysis attacks and timing attacks in practical use. Fig.5 and Fig.7 demonstrate our hypothesis.

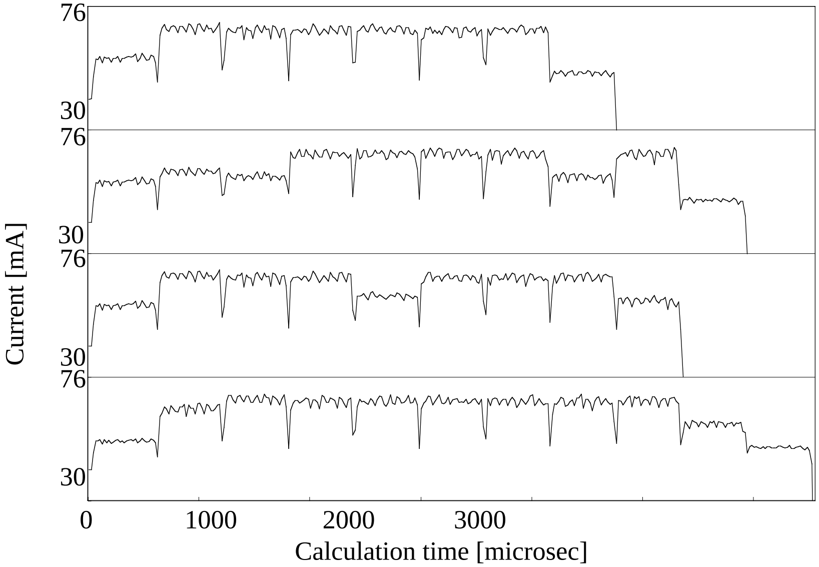

Fig. 7. Current Waveform of OPM

OPM resists to side channel attacks in practical use. However, further enhancement is possible. We add DC (see section 4) when double modular exponentiation calculations are performed. This DC forces the operation of all three modular multiply units. If a calculation requires the use of only two units, DC is performed in the unused unit. Fig.5 shows the DC via oblique lines.

Fig.8 shows the current waveform resulting from three types of double modular exponentiations. We compared OPM+DC, D-RLC+DC and D-LRC+DC. It is difficult to distinguish between "0" and "1". Three methods show the same current waveform each for every exponent.

Table 4 shows the results of three methods where the base and modulus are 1024 bits each and the exponents are 8 bits. The values of OPM were obtained from the average of seven patterns for each B1 and B2, 8080, 80ff, f0f0, aaaa, f0ff, aaff, ffff (hexadecimal digit). Followings are indicated from Table.4:

- OPM shows the best current consumption of double exponentiations,
- OPM shows fairly good characteristics of number of gates and calculation time compares best other method

For OPM, we anticipated an increase in the number of gates required since gate requirements are proportional to the number of modular multipliers (we use three modular multiply units). But OPM needed only about 10% more gates due to the total circuit scale expansion from the addition of control circuits, etc. when compared with the total circuit scale.

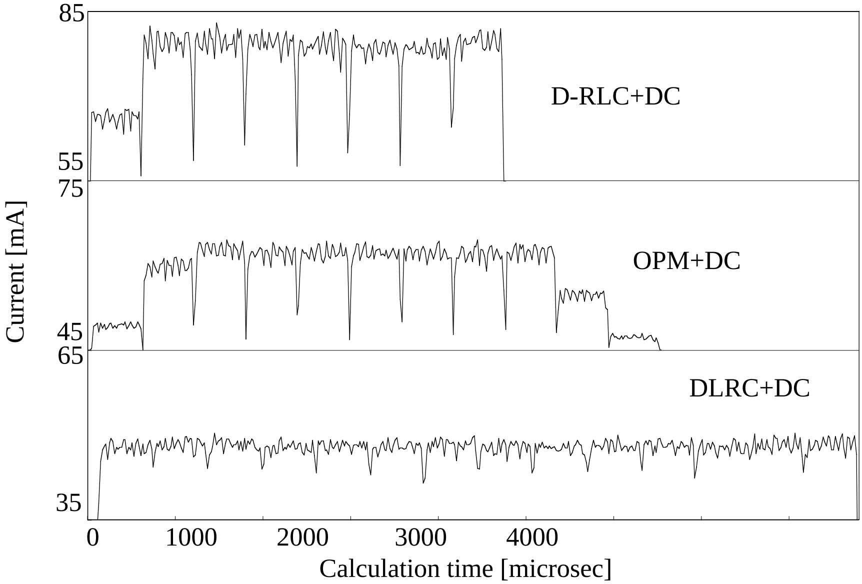

Fig. 8. Current Waveform of D-LRC+DC, D-RLC+DC and OPM+DC

Although the average current consumption is higher and more gates are required, OPM has the advantages of reduced calculation time and lower power consumption.

The coprocessor using OPM featuring high speed, low power consumption, small size and resistance to power analysis attacks and timing attacks are ideal for mobile telecommunication terminals.

Table 4. Overall Comparison between OPM, D-RLC+DC and D-LRC+DC

	Number of gate [gates]	Average calculation time [microsec]	Current consumption of double exponentiations [microsec*mA]
OPM	62367 (1.1)	2723 (0.62)	154375 (0.74)
D-RLC+DC	74164 (1.31)	2390 (0.54)	182065 (0.87)
D-LRC+DC	56688 (1)	4400 (1)	209106 (1)

*(….) shows relative values

By replacing from modular multiply to add on elliptic curve, the concept of OPM could be used in elliptic curve cryptosystems.

6 Conclusion

Our coprocessor design features the following characteristics:

- simultaneous double modular exponentiations performed at high speed within practical time
- small size and low power consumption
- resistance to side channel attacks

This coprocessor provides all of these well-balanced characteristics, making it ideal for mobile telecommunication terminals.

References

1) "Digital Signature Standards", Federal Information Processing Standard publication X, 1993 February 1
2) K.Nyberg, A.Rueppel, "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem", Advanced in Cryptology-EUROCRYPT' 94, Springer-Verlag.
3) R.Cramer, V.Shoup, "A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack", Lecture Note in Computer Science. Advanced in Cryptology-CRYPTO'98, Springer-Verlag, pp.13-25.
4) J.Anzai, N.Matsuzaki, T.Matsumoto, "A Quick Group Key Distribution Scheme with Entity Revocation", Advanced in Cryptology-ASIACRYPTO’99, Springer-Verlag, pp.333-347.
5) N.Matsuzaki, J.Anzai, T.Matsumoto, "Light Weight Broadcast Exclusion using Secret Sharing", Fifth Australasian Conference on Information Security and Privacy, Springer-Verlag, pp. 313-327.
6) P.C.Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems", Advanced in Cryptology-CRYPTO'96, Springer-Verlag, pp.104-113.
7) T.S.Messerges, E.A.Dabbish, R.H.Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards", Cryptographic Hardware and Embedded Systems-CHES'99, Springer-Verlag, 1999, pp.144-157.
8) L.Goubin, J.Patarin, "DES and Differential Power Analysis : The Duplication Method", Cryptographic Hardware and Embedded Systems-CHES'99, Springer-Verlag, 1999, pp.158-172.
9) H.Handschuh, P.Paillier, J.Stern, "Probing Attacks on Tamper-Resistant Devices", Cryptographic Hardware and Embedded Systems-CHES'99, Springer-Verlag, 1999, pp.303-315.
10) A.Andreasyan, G.Khachatrian, "New Double Exponentiation Algorithms", Third International Workshop on practice and Theory in Public Key Cryptography PKC2000, The Poster Papers Collection p.9-15, ISBN 0-73262-130-5, Monash Univ.
11) A.J.Menezes, P.C.Oorchot, S.A.Vanstone, "HANDBOOK of APPLIED CRYPTOGRAPHY", CRC press, pp.614-615, pp.620-627.