Fast Blum-Blum-Shub Sequence Generation Using Montgomery Multiplication

M.G.Parker,

University of Bergen, Department of Informatics, University of Bergen,

N-5020 Bergen, Norway, matthew@ii.uib.no.

A.H.Kemp,

Institute of Integrated Information Systems, School of Electronic and Electrical Engineering, University of Leeds, LS2 9JT Leeds, UK.

S.J.Shepherd,

Telecommunications Research Centre, Department of Electronic and Electrical Engineering, University of Bradford, Bradford, BD7 1DP, UK.

Abstract

VLSI modules are proposed for fast, efficient generation of high-throughput Blum-Blum-Shub (BBS) and BBS-like sequences using Montgomery Multiplication, where post-processing associated with Montgomery's algorithm can be eliminated.

1 Introduction

Public key cryptosystems ensure secrecy between communicating parties without the need to distribute secret keys. The most famous public key cryptosystem is that devised by Rivest, Shamir, and Adleman (RSA) [1]. Another lesser-known public key cryptosystem is the Quadratic Residue Cipher (QRC) introduced by Blum, Blum, and Shub [2], which relies on the ease of squaring an integer, mod n, as compared to the intractability of finding the square root of a number, mod n when n is large. As with RSA the valid recipient publishes the prime factors of n, where $n = pq$ and p and q are strong primes. The sender scrambles his message with a bit sequence (Blum-Blum-Shub Sequence, BBS) comprised of the concatenation of the least significant bits of a series of successive squares, mod n, starting with a randomly chosen 'seed', and then appends the final value in the sequence of successive squares to the scrambled message. Only the valid recipient can unscramble the data as only he/she knows the factors of n, and knowing p and q allows the recipient to find the starting seed, i.e. the 2^nth root, mod n, using the final successive square which was appended to the received sequence. Once the starting seed is known the recipient can regenerate the scrambling sequence and decrypt the ciphertext. As with RSA the scheme relies on the inability to factor n when $n = pq$, and p and q are large strong primes. The advantages of QRC over RSA are that RSA is a deterministic cipher, whereas QRC is probabilistic because it starts from a randomly chosen seed. Moreover it is known that RSA can leak partial information about the message sent, whereas no such weakness is known for QRC. Also it is a bit easier to generate BBS for QRC, than successive exponents for RSA. Finally, spectral randomness properties of BBS make it very suitable for use as a spreading sequence in a spread-spectrum communications system, thereby providing the possiblity of high-security spread spectrum [3]. QRC can also provide digital signature and resistance to a chosen ciphertext attack, but at greater cost than with RSA [4]. Detailed comparisons of QRC and RSA are given in [5, 4].

Although implementation complexity of QRC is slightly less than RSA, it is still costly. Successful implementation of QRC relies on efficient generation of BBS. This paper proposes novel hardware to allow highly efficient generation of BBS using Montgomery Multipliers (MMs) [6, 7, 8, 9]. MMs are particularly suited to VLSI implementation of modular multiplication as they allow computation of modular reduction to begin before computation of the most-significant-bit has been completed. This speeds up successive modular arithmetic operations (such as squaring). However the drawback is the multiplicative offset associated with MM. In this paper this offset is incorporated into BBS generation without cost, and further simplification is made possible by considering the generation of BBS-like sequences.

2 Blum-Blum-Shub Sequence (BBS) and QRC

Let $n = pq$, where p and q are primes satisfying $p = 4k_p + 3$, $q = 4k_q + 3$. The BBS [2, 5] of integer a, mod n, is given by,

$$\begin{aligned} \mathrm{BBS}_n(a) &= L(\mathrm{BS}_n(a)) \quad \text{where} \\ \mathrm{BS}_n(a) &= (a, a^2, a^4, \ldots, a^{2^{t-1}}), \qquad \bmod n, \qquad t \leq \mu_n \end{aligned} \tag{1}$$

where the order[i] of $\mathrm{BBS}_n(a)$ is $\mu_n = \mathrm{lcm}^{\mathrm{ii}}\ (\mu_p, \mu_q)$, $\mu_p = \ \mathrm{ord}\ _{\mathrm{ord}_{p(a)}}(2)$, $\mu_q = \ \mathrm{ord}\ _{\mathrm{ord}_{q(a)}}(2)$, and L means 'concatenate the h least-significant-bits (lsbs) of each successive residue'. (1) requires the following gcd[iii] condition to be satisfied,

$$\text{Condition:} \qquad \gcd(2,\ \mathrm{ord}_p(a)) = \gcd(2,\ \mathrm{ord}_q(a)) = 1 \tag{2}$$

For a good choice of k_p and k_q, approximately $\frac{1}{4}$ of all integers, a, satisfy Condition (2), and can generate BBS [2]. Condition (2) can be virtually guaranteed by selecting a random integer $\alpha < n$. Then $a = \langle \alpha^2 \rangle_n$ nearly always satisfies (2). The randomness and cryptographic strength of the QRC are fatally compromised if the BBS has short period, μ_n, or if n can be factored. Therefore, ideally p and q are 'strong' primes, in other words p, q are chosen such that $p, q, k_p, k_q, 2k_p + 1, 2k_q + 1$ are all prime. This ensures a very large BBS period of $k_p k_q$ or $2k_p k_q$, and also maximises the difficulty of factoring n.

[i] $\mathrm{ord}_f(e)$ - 'the order of e, mod f, is t, where $\langle e^t \rangle_f = 1$, $\langle e^s \rangle_f \neq 1$ for some positive integers s, t, $s < t$'

[ii] lcm(.) - 'lowest common multiple of'

[iii] gcd(.) - 'greatest common denominator of'

The QRC generates ht BBS bits and XOR's them with ht bits of the message. Finally it appends the complete final residue, $\left\langle a^{2^t}\right\rangle_n$, to the XOR'ed message, and the bit string is transmitted. No polynomial time algorithm exists to regenerate BBS from knowledge of the final residue $\left\langle a^{2^t}\right\rangle_n$ and n [10]. The message is therefore secure from eavesdroppers.

[2] shows that concatenating $h = 1$ lsbs from each residue of $\mathrm{BS}_n(a)$ is provably secure. Further analysis [11, 10] indicates a lower bound of $h = \log_2(\log_2(n))$ bits which can be extracted from each residue. However even this is a lower bound and it is an open question as to how large h can be. The relevance to this paper is that BBS generation speed rises linearly with h.

3 Montgomery Multiplication (MM)

MM [7] efficiently computes,

$$c = \left\langle abr^{-m-1}\right\rangle_n + jn \qquad j \in \{0, 1\} \tag{3}$$

where $0 < a < n$, $0 < b < 2n$, r is the radix of representation for a, b, and c, $\gcd(r, n) = 1$, and $m =$ [iv] $\lceil \log_r(n) \rceil$. MM can be used for squaring if $b = a$, requiring two successive MMs,

$$\begin{aligned} &1. \quad c = \langle aar^{-m-1}\rangle_n + j_1 n \\ &2. \quad \langle a^2\rangle_n + j_2 n = \langle cdr^{-m-1}\rangle_n + j_2 n, \qquad \text{where } d = \langle r^{2m+2}\rangle_n, \qquad j_1, j_2 \in \{0, 1\} \end{aligned} \tag{4}$$

[iv] $\lceil . \rceil$ - 'round up to the next highest integer'

MM is inefficient for squaring as two successive MMs are required. This is also true for direct generation of BBS using MM. Moreover, the result may be offset by $j_2 n$, $j_2 \in \{0, 1\}$.

4 Fast BS Generation Using MM

Consider the sequence generated by successive applications of MM,

$$c_i = \left\langle c_{i-1} c_{i-1} r^{-m'} \right\rangle_n + j_i n, \qquad j_i \in \{0, 1\} \tag{5}$$

where $0 < c_i < 2n$ and $m' = m+2$. In the previous section $m' = m+1$. This is because inputs a and b were specified with a maximum of m and $m+1$ radix-r digits, respectively, where the $m + 1^{\mathrm{th}}$digit of a is zero and used to reduce the output, c, back to $m + 1$ digits. However, in this section both inputs, c_{i-1}, require $m + 1$ digits each. Therefore our MM for successive squaring requires $m' = m + 2$ rows, where row $m + 2$ reduces the output, c_i, back to $m + 1$ digits. Let $c_0 = \left\langle r^{m'} a \right\rangle_n$. Then the sequence generated by (5), $W_n(c_0) = \{c_i : 0 \leq i < t\}$, is given by,

$$W_n(c_0) = \left\langle r^{m'} (a, a^2, a^4, \ldots, a^{2^{t-1}}) \right\rangle_n + (0, j_1 n, j_2 n, \ldots, j_{t-1} n) \qquad j_i \in \{0, 1\} \tag{6}$$

where the '+' indicates vector addition. If a satisfies (2) then,

$$W_n(c_0) = \left\langle r^{m'} \mathrm{BS}_n(a) \right\rangle_n + (0, j_1 n, j_2 n, \ldots, j_{t-1} n) \tag{7}$$

As before, for a good choice of k_p and k_q, approximately $\frac{1}{4}$ of all integers, c_0, generate a, such that (2) is satisfied, on condition that $\gcd(r^{-m'}, n) =$

1, which is virtually guaranteed, and is certain for $r = 2$. (In any case, $\gcd(r, n) = 1$ is a requirement for the operation of MM).

To generate $\mathrm{BS}_n(a)$ using $W_n(c_0)$, one again performs two MMs:

$$\text{Step 1. } c_i = \left\langle c_{i-1}c_{i-1}r^{-m'} \right\rangle_n + j_i n \qquad \text{Step 2. } a^{2^{i-1}} = \left\langle c_{i-1}1r^{-m'} \right\rangle_n \tag{8}$$

The result of Step 2 always gives $0 \leq a^{2^i} < n$, $\forall i$ (Appendix A). Moreover, Step 2 can be performed in parallel with the subsequent Step 1, (unlike (4) where the operations are sequential and both in the squaring loop). A similar idea to (8) is suggested in [7, 8, 9], but there it is with reference to RSA exponentiation. In contrast we propose (8) for successive squaring. Finally, for $r = 2$, and assuming it is required to compute only the [v] $h = \lfloor \log_2(m) \rfloor$ lsbs of each member of $\mathrm{BS}_n(a)$, a reduction in latency for Step 2 approaching $\frac{1}{2}$ is possible, as is a similar hardware reduction for a fully parallel implementation of Step 2. (For QRC this last simplification is not possible as one needs to append the full final residue to the bit stream). As one of the inputs to the second MM is fixed at $'1'$, further hardware simplification to Step 2 is possible, for both parallel and serial versions. Fig 1 shows fully parallel and parallel-serial architectures for this algorithm. The fully parallel version is only useful if m' different BS sequences are interleaved. For both parallel and serial, c_i becomes available at the output of MM1, LSB first, and is fed back to the input of MM1 immediately, giving a throughput rate of one new member of sequence $W_n(c_0)$, (and $\mathrm{BS}_n(a)$),

[v] $\lfloor . \rfloor$ - 'round down to the next lowest integer'

every m' clock cycles. The first member of $\mathrm{BS}_n(a)$ begins to appear after $2m'$ clock cycles.

Assuming a conventional modular multiplier is $\simeq 2$ times slower than a MM, then BBS generation using MM is $\simeq 2$ times faster than conventional generation. This is because Step 1 does not wait for Step 2 to finish between iterations.

5 Improvements

For $r = 2$, $\lfloor \log_2(m) \rfloor$ lsbs of each member of $\mathrm{BS}_n(a)$ are mutually uncorrelated with each other and with the same bits of any other member of the same sequence [10, 11]. This assumes $0 \leq a^{2^i} < n$, $\forall i$, mod n. For $W_n(c_0)$, the c_i satisfy $0 \leq c_i < 2n$, and the sequence is a constant multiple, $(\times r^{m'})$ of $\mathrm{BS}(a)$, mod n. One can therefore compute the sequence, $T_n(d_0)$, where $d_i = \langle c_i \rangle_n$, i.e. the d_i satisfy $0 \leq d_i < n$, $d_i = \left\langle r^{m'} a^{2^i} \right\rangle_n$. $T_n(d_0)$ is generated by replacing Step 2 of (8) with,

$$\begin{aligned} \text{Revised Step 2.}\ & d_{i-1} = c_{i-1} \\ & \text{if } d_{i-1} > n, d_{i-1} = d_{i-1} - n \end{aligned} \tag{9}$$

and for $r = 2$ the $\lfloor \log_2(m) \rfloor$ lsbs of $T_n(d_0)$ will again be mutually uncorrelated. For large m, (9) requires less hardware and is faster than Step 2 of (8).

The sequence throughput remains unchanged, whether the sequence be $\mathrm{BS}_n(a)$ or $T_n(d_0)$. To substantially increase throughput, the parallel form of Step 1 (and Step 2) can interleave m' different BS or T sequences, originating from m' different seeds and/or moduli, $c_{0,k}$ and n_k, respectively. In this way, $\lfloor \log_2(m) \rfloor$ lsbs can be appended to the output bit-stream every clock cycle. An argument for the maintained cryptographic strength of interleaved BS or T sequences is given in Appendix B. Such interleaving dramatically increases throughput.

Finally, for the user who knows p and q, $\mathrm{BS}_p(a)$, $\mathrm{BS}_q(a)$, or $T_p(d_0)$, $T_q(d_0)$ sequences can be generated in parallel, mod p and mod q, with a final combining of sequence residues outside the loop to generate $\mathrm{BS}_n(a)$, or $T_n(d_0)$, using the Chinese Remainder Theorem [13], thereby reducing area four times, and halving sequence generation time.

6 Conclusion

Montgomery Multiplication (MM) is appropriate for fast generation of BBS Sequences in spite of the constant multiplicative offset inherent within successive squarings using MM. Post-processing associated with MM is taken out of the squaring loop to occur in parallel with the squaring. Further area/time savings are achieved by retaining the multiplicative offset to the BS Sequence whilst ensuring each member of the sequence is less than the modulus. High-throughput sequence generation is obtained using interleaved squarings and fully parallel MM. As a final suggestion one could eliminate post-processing completely by simply generating the concatenation of $\lfloor \log_2(m) \rfloor$ lsbs of each member of $W_n(c_0)$. In this way Step 2 is eliminated completely. This sequence is probably also mutually uncorrelated, although we have no proof for this. The hardware improvements presented in this paper enhance the proposal to use the Quadratic Residue Cipher in cryptosystems and secure spread-spectrum systems.

7 Acknowledgements

This work was supported in part by EPSRC grants GR/K48914,GR/K06839, and by NFR Project Number 119390/431.

8 Appendix A - To Verify Step 2 of (8)

From [7], the output from MM2 of Fig 1 is given by,

$$\left\langle a^{2^i} \right\rangle_n + vn = (c_i + Qn)/r^{m'} \tag{10}$$

where v is a small positive integer. The maximum output from MM2 occurs with worst-case c_i and Q both equal to $r^{m'} - 1$, respectively, so,

$$\left\langle a^{2^i} \right\rangle_n + vn \leq \left(\frac{r^{m'} - 1}{r^{m'}} \right) (1 + n) \tag{11}$$

As $\left\langle a^{2^i} \right\rangle_n \neq 0$ for any valid seed, a, (11) implies $v = 0$, i.e.,

$$(c_i + Qn)/r^{m'} < n \qquad \forall i \qquad \blacksquare \tag{12}$$

9 Appendix B - To Justify the Cryptographic Strength of Interleaved BBS and T Sequences

Let the sender generate a BBS sequence, mod n, of length ht bits using a randomly generated starting seed a and its $t - 1$ successive squares, mod n.

The strength of the QRC partially rests on the impossibility of an eavesdropper guessing a starting seed b such that $\left\langle b^{2^i} \right\rangle_n \in \{\left\langle a^{2^j} \right\rangle_n, 0 \leq j < t\}$, $0 \leq i < t$ in polynomial (in m) time using polynomial resources. Therefore $m' = m + 2$ eavesdroppers will have no more success than one eavesdropper. This is the same as saying that m' interleaved BBS sequences, mod n, will be mutually uncorrelated. This argument, as with single sequence BBS, is conditional on a sufficiently large sequence order $k_p k_q$ or $2k_p k_q$. Similar arguments can be used when using a different modulus, n_k, for each interleaved sequence, both for BBS and T sequences. $\blacksquare$

References

[1] Rivest,R.L., Shamir,A., Adleman,L.: "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Comm.ACM*, Feb 1978, **21**,(2), pp. 120-126

[2] Blum,L., Blum,M., Shub,M.: "A Simple Unpredictable Pseudo-Random Number Generator", *SIAM J. Comput*, May 1986,**15**,(2), pp. 364-383

[3] Shepherd,S.J., Kemp,A.H., Barton,S.K.: "An Efficient Key Exchange Protocol for Cryptographically Secure CDMA Systems," *Globecom '96, London, U.K.*, Nov 1996

[4] Rubin,F.: "The Quadratic Residue and Double Quadratic Residue Ciphers", *Cryptologia*, 1995, **IX**,(3), pp. 275-28

[5] Shepherd,S.J., Sanders,P.W., Stockel,C.T.: "The Quadratic Residue Cipher and Some Notes on Implementation", *Cryptologia*, July 1993, **XVII**,(3), pp. 264-282

[6] Montgomery,P.L.: "Modular Multiplication without Trial Divison", *Math. Computation*, 1985, **44**, pp. 519-521

[7] Walter,C.D.: "Systolic Modular Multiplication," *IEEE Trans. on Computers*, March 1993, **42**, (3), pp. 376-378

[8] Kornerup,P.: "A Systolic, Linear Array Multiplier for a Class of Right-Shift Algorithms," *IEEE Trans. on Computers*, August 1994, **43**, (8), pp. 892-898

[9] Blum,T.,Paar,C.: "Montgomery Modular Exponentiation on Reconfigurable Hardware", *IEEE Symp. on Computer Arithmetic*, Adelaide, 1999

[10] Vazirani,U.V., Vazirani,V.V.: "Efficient and Secure Pseudo-Random Number Generation", *Proc 25th IEEE Sym. Foundations of Computer Science, '84*, 1984, pp. 458-463

[11] Yao,A.C.: "Theory and Applications of Trapdoor Functions", *Foundations of Computer Science, IEEE 23rd Symp*, 1982, pp. 80-91

[12] Kemp,A.H., Shepherd,S.J., Barton,S.K.: "Correlation Properties of a Class of Cryptographically Secure Spreading Sequences," *ISSSTA '96, Mainz, Germany*, Sept 22-25, 1996

[13] McClellan,J.H., Rader,C.M.: **Number Theory in Digital Signal Processing,** Prentice Hall, 1979

Figure 1: Parallel and Serial-Parallel Blum Sequence Generators Using Montgomery Multipliers

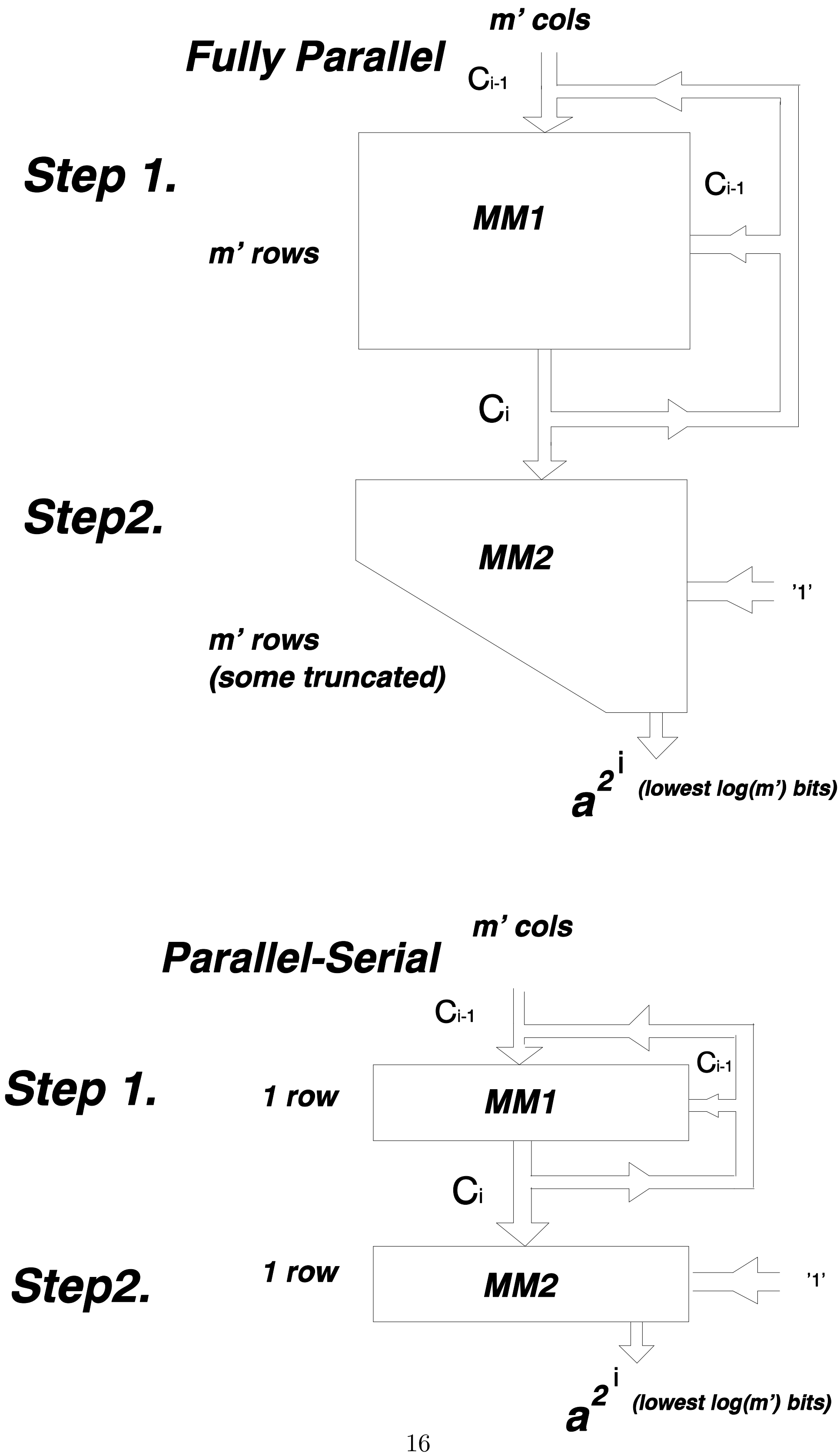

Figure 1: Parallel and Serial-Parallel Blum Sequence Generators Using Montgomery Multipliers