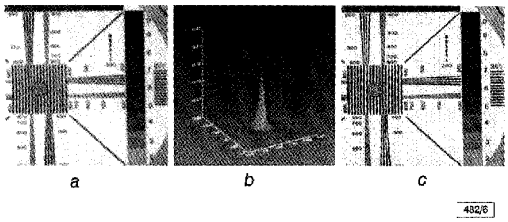

Fig. 6 *Real 256 × 256 photographically out-of-focus blurred image, estimated 2D PSF, and restored image*

a Real image
b Estimated 2D PSF
c Restored image

Conclusions: We propose a fully digital auto-focusing system based on novel out-of-focus blur estimation and restoration algorithms. The main advantages of the proposed PSF estimation algorithm are that it can estimate both radius and sample values of an arbitrary circularly symmetric blur, and that it does not require the DFT or a numerical optimisation process for parameter estimation. As a result it is far more efficient than the existing methods in terms of computational complexity. The proposed system can accurately estimate not only simulated out-of-focus blur, but also real photographical blur.

© IEE 1998
5 December 1998
Electronics Letters Online No: 19980762

Sang Ku Kim and Joon Ki Paik (*Department of Electronic Engineering, Chung-Ang University, 221 Huksuk-Dong, Tongjak-Ku, Seoul 156-756, Korea*)

E-mail: paikj@chungang,edu

References

1 KIM, S.K., KIM, T.K., and PAIK, J.K.: 'A fully digital auto-focusing system based on image restoration'. Proc. 1997 IEEE Region10 Annual Conf., December 1997, Vol. 1, pp. 13–16
2 JAIN, A.K.: 'Fundamentals of digital image processing' (Prentice-Hall, 1989)
3 CHUNG, Y.C., and PAIK, J.K.: 'A fast adaptive image restoration filter for reducing block artifact in compressed images', *IEEE Trans. Consum. Electron.*, 1997, **CE-43**, (4), pp. 1340–1346
4 ANDREWS, H.C., and HUNT, B.R.: 'Digital image restoration' (Prentice-Hall, 1977)

Batch verifying multiple RSA digital signatures

L. Harn

A digital signature is analogous to an ordinary hand-written signature used in signing messages. RSA digital signatures have been adopted by Visa and Mastercard in the secure electronic transactions (SET) standard for providing security of electronic transfers of credit and payment information over the internet. In SET, signatures are used to provide certificates for public keys and to authenticate messages. The authors propose an efficient method of verifying RSA digital signatures. Instead of verifying one signature at a time, it is proposed to batch verify RSA signatures simultaneously. This approach maintains the same computational load as verifying a single signature. Thus, a significant reduction in signature verification time can be achieved.

Introduction: A digital signature is analogous to an ordinary hand-written signature used for signing messages. It must be unique and private to the signer. More specifically, suppose that B is the recipient of a message m signed by A. Then, A's signature must satisfy three requirements [1]:

(i) B must be able to validate A's signature on m easily
(ii) It must be impossible for anyone, including B, to forge A's signature
(iii) It must be possible for a judge or third party to resolve any dispute between A and B.

At this time, there are two popular public-key algorithms which can provide digital signatures: the RSA scheme [2], in which the difficulty of breaking the scheme is based on solving the factoring of a large integer into two large prime factors; and the ElGamal scheme [3], in which the difficulty of breaking the scheme is based on solving the discrete logarithm problem.

The RSA digital signature has been adopted by Visa and Mastercard in the secure electronic transactions (SET) standard [4] for providing security of electronic transfers of credit and payment information over the internet. In SET, signatures are used to provide certificates for public keys and to authenticate messages. Since public key cryptography requires intensive computation, it is desirable to speed up these public key computations by using either special-purpose hardware or efficient software algorithms. In this Letter, we propose an efficient algorithm to batch verify RSA signatures signed by the same private key. Instead of verifying one signature at a time, we propose to verify multiple signatures simultaneously. This approach maintains the same computational load as is required to verify a single signature. Thus, a significant reduction in signature verification time can be achieved. The application of our scheme can be used in electronic commerce to significantly reduce the computational load. For example, multiple public-key certificates signed by the same certificate authority (CA), or multiple authenticate messages signed by the same payment gateway, can be verified based on our scheme.

Algorithm: The scheme is based on the property that RSA signatures satisfy the multiplicative property. The following example illustrates this property.

Assume that the RSA signature scheme has selected the modulus $n = pq$, where p and q are two large secret primes. Let us denote e as the public key and d as the private key, where $ed \bmod (p-1)(q-1) = 1$, and $h(.)$ is a one-way hash function. The RSA signature of a message m_i is $S_i = h(m_i)^d \bmod n$. The signature verification is performed by checking whether $h(m_i) = S_i^e \bmod n$. We assume that the RSA signatures of $m_1, m_2, ..., m_t$ are $S_1, S_2, ..., S_t$. Due to the multiplicative homomorphism, we can easily prove the following theorem.

Theorem 1: If the signatures $S_1, S_2, ..., S_t$, of $m_1, m_2, ..., m_t$, are all valid, then we have $(\pi_{i=1}^{t} S_i)^e = \pi_{i=1}^{t} h(m_i) \bmod n$.

We call the product of individual signatures $\pi_{i=1}^{t} S_i \bmod n$ the 'multiplicative signature' of $m_1, m_2, ..., m_t$. To verify this multiplicative signature, we need only to compute one exponentiation. The objective of this Letter is to prove that the multiplicative signature is a valid signature of messages $m_1, m_2, ..., m_t$.

Theorem 2: If the multiplicative signature satisfies $(\pi_{i=1}^{t} S_i)^e = \pi_{i=1}^{t} h(m_i) \bmod n$, then we say that the multiplicative signature is a valid signature of messages $m_1, m_2, ..., m_t$.

Proof: A valid signature should satisfy three properties, as stated in the Introduction. Due to its homomorphic property, the multiplicative RSA signature can be easily verified according to Theorem 1. Thus, it satisfies the first requirement.

Since the multiplicative signature is the product of all individual signatures and it requires a secret key to generate all these individual signatures, any arbitrator is able to resolve any dispute between the signer and the verifier. Thus, it satisfies the third requirement.

The only concern is whether an attacker (including the verifier) can forge a valid multiplicative signature without knowing the secret key. If the signer signs each message directly without applying the one-way hash function, there is a possible attack to forge a valid multiplicative signature. The attacker can ask the signer to sign each message m_i individually such that $M = \pi_{i=1}^{t} m_i \bmod n$, and then the multiplicative signature becomes the valid signature of the message M. However, incorporating the one-way hash function into the signature scheme, it is computationally infeasible (i.e. even for the signer) to obtain messages $m_1, m_2, ..., m_t$, such that $h(M) = \pi_{i=1}^{t} h(m_i) \bmod n$.

Thus, we conclude that the multiplicative RSA signature is a valid signature for all individual messages. *Q.E.D.*

Authorized licensed use limited to: University of Missouri System. Downloaded on March 23, 2009 at 14:41 from IEEE Xplore. Restrictions apply.

Note that the signer can generate two individual signatures $S_1' = S_1/2$ and $S_2' = 2S_2$ such that their product satisfies $S_1' S_2' = S_1 S_2$. Since this ability can only be associated with the signer with a proper secret key, any related dispute can be easily solved. In addition, in case the batch verification fails, all individual signatures must be verified separately.

Conclusion: We have proposed an efficient way to batch verify RSA signatures simultaneously. The performance of our scheme is almost constant.

© IEE 1998
7 April 1998
Electronics Letters Online No: 19980833

L. Harn (*Racal Datacom Group, Sunrise, FL 33323-2899, USA*)

References

1 DENNING, D.E.: 'Cryptography and data security' (Addison-Wesley, Reading, MA, 1982)
2 RIVEST, R.L., SHAMIR, A., and ADELMAN, L.: 'A method for obtaining digital signatures and public-key cryptosystem', *Commun. ACM*, 1978, **21**, (2), pp. 120–126
3 ELGAMAL, T.: 'A public-key cryptosystem and a signature scheme based on discrete logarithms', *IEEE Trans. Inf. Theory*, 1985, **IT-31**, pp. 469–472
4 SET Specification, http://www.visa.com/cgibin/vee/ht/ecomm/set/downloads.html?2+0#specs

Cryptanalysis of 'Labyrinth' stream cipher

S.R. Blackburn, K. Brincat, F. Mirza and S. Murphy

Cryptanalysis of the stream cipher 'Labyrinth', a cipher recently proposed by Lin and Shepherd, is performed. Given only 2^{30} known bits of keystream, the 119 bit key of Labyrinth is recovered in under a second of computation using a DEC Alpha.

Introduction: The stream cipher 'Labyrinth' has recently been proposed by Lin and Shepherd [2]. They suggest using Labyrinth in bulk data encryption applications, in particular for ATM packet-switching networks. The cipher has a 119 bit key, and Lin and Shepherd report encryption speeds of 1.6 Gbits^{-1} when the cipher is implemented in hardware.

This Letter contains a cryptanalysis of Labyrinth. This requires approximately 2^{30} bits of known keystream (~1 s of the cipher's output); simulations indicate that a DEC Alpha would recover the key of the cipher in < 1 s using this attack.

Cipher: This section briefly describes Labyrinth. There are some misprints and ambiguities in the original paper [2]. Three misprints are: (i) Register A should have feedback polynomial $x^{43} + x^6 + x^4 + x^3 + 1$ (the polynomial given in the paper is not primitive); (ii) In the definition of Register B, the equations should read $y_i = x_{i-1} \oplus (x_{i-1} \gg 15)$ and $x_i = y_i \oplus (y_i \ll 17)$; (iii) The first line of the description of the key controlled hash function in Section 2.4 should read $Y = (U \oplus V, V)$. For this reason we have additionally relied on a software simulation of the algorithm (written in C, and kindly provided by B. Lin) in producing this description.

Labyrinth may be thought of as comprising two main components. The first component is a linear finite state machine; its current state may be thought of as a binary vector v of length 107, the next state is obtained by multiplying the current state by a fixed (public) 107×107 invertible matrix M. The output of the finite state machine consists of a binary vector r of length 48. This vector is obtained from the current state v by calculating $r = \phi(v)$, where ϕ: $\mathbb{Z}_2^{107} \rightarrow \mathbb{Z}_2^{48}$ is a fixed (public) linear function. The initial state of the finite state machine is controlled by 96 bit of the key – these bits determine 96 of the 107 components of the initial state v_0 of the machine; the remaining components of v_0 are key independent (and public).

The second component is a key- and time-dependent non-linear function f. The 23 key bits of the cipher not involved in determining the initial state of the linear finite state machine are used to determine this non-linear function. The function f takes the 48 bit vector r produced by the finite state machine as input and produces a 32 bit output. We describe f in more detail below.

The 23 bits of key which influence f are broken up into three parts. The first part determines an odd integer t such that $1 \leq t \leq 31$ (and so this part consists of 4 bits). The second part determines an odd integer p such that $1 \leq p \leq 15$ (and so this part consists of 3 bits). The remaining 16 bits of key material determine a 16 bit string K. Given t, p, K and the 48 bit input vector r, the output of f at time i is calculated as follows. The 48 bit vector r is regarded as eight 6 bit quantities $r_1, r_2, \ldots, r_8$. For each n such that $1 \leq n \leq 8$ r_n is fed into the nth S-box as detailed in the data encryption standard (DES) [1]; each S-box produces a 4 bit output x_n. The outputs x_n are concatenated to form a 32 bit quantity x. The 16 most significant bits of x are modified by XORing them with the 16 least significant bits of x, to produce the 32 bit quantity y. The quantity y is then rotated t bits to the left to form the 32 bit quantity y'; we regard y' as a 32 bit integer. Let K_i' be the 16 bit quantity formed by rotating K a total of ip times. This is the only time dependent part of the function f; note that $K_i' = K_{i+16}'$ for all i. The output of f is defined to be the sum modulo 2^{32} of y' and the 32 bit integer l whose 16 least significant bits are equal to the 16 most significant bits of y' and whose 16 most significant bits are equal to K_i'.

The cipher operates as follows. The initial state v_0 of the linear finite state machine is determined by 96 bits of the key. At time i, the cipher outputs a block of 32 bits of keystream given by $f(\phi(vM^i))$, where f depends on the remaining 23 bits of key material and on the time i. The keystream block is then XORed with a 32 bit plaintext block in the classic stream cipher fashion.

Cryptanalysis: We cryptanalyse the cipher in two stages. In stage I, we obtain most of the key material that is used to control the non-linear function f. More precisely, we obtain the 16 bit string K and the integer p by using the fact that for a fixed key the function f is not quite surjective. We then guess the 4 bits of the key that determine the integer t used in the function f. In Stage II, we use outputs of the S-box stage of the function f to determine a collection of linear relations that the initial state v_0 of the linear finite state machine should satisfy. If our guess for the integer t in stage I was correct, these linear relations determine the initial state v_0 (and hence the remainder of the key). If our guess for the integer t was incorrect, the linear relations quickly become inconsistent and so we repeat the process with a different guess.

Stage I: Key material controlling the non-linear function: We recover the key material associated with the quantities K and p by using the following lemma.

Lemma 1 Let A and B be the 16 bit integers formed respectively from the most significant and least significant 16 bits of the 32 bit output of f at time i. If $B \neq 2^{16} - 1$, then the quantity K_i' used in the computation of f has the property that

$$K_i' \neq A - B - 1 \bmod 2^{16}$$

Proof: Let W and Z be the integers formed respectively from the most significant and least significant 16 bits of the quantity y' given in the description of f. The final stage of the calculation of f gives us, for some $\epsilon \in \{0, 1\}$,

$$2^{16}K_i' + W + 2^{16}W + Z = 2^{32}\epsilon + 2^{16}A + B$$

Suppose for a contradiction that $K_i' = A - B - 1 \bmod 2^{16}$, so for some $\delta \in \{0, 1\}$, $K_i' = A - B - 1 + \delta 2^{16}$. Then

$$Z = \{(2^{16} - 1)(\epsilon - \delta) + (B - W + 1)\}(2^{16} + 1) + (\epsilon - \delta - 1)$$

If $\epsilon = \delta$, then $Z = -1 \bmod (2^{16} + 1)$, a contradiction.

If $\epsilon = 1$, $\delta = 0$, then $Z = 0 \bmod (2^{16} + 1)$, so $Z = 0$. Thus $2^{16} + B - W = 0$, and so $W - B = 2^{16}$, a contradiction.

If $\epsilon = 0$, $\delta = 1$, then $Z = -2 \bmod (2^{16} + 1)$, so $Z = 2^{16} - 1$. Thus $\{B - W - (2^{16} - 1)\}(2^{16} + 1) = 0$, so $B - W = 2^{16} - 1$. Therefore $W = 0$ and $B = 2^{16} - 1$, a contradiction.

Since all three cases lead to a contradiction, $K_i' \neq A - B - 1 \bmod 2^{16}$ when $B \neq 2^{16} - 1$, as required.

The rotation K_i' of K is used in the production of every 16th output block. If we therefore decimate an output stream of blocks

Authorized licensed use limited to: University of Missouri System. Downloaded on March 23, 2009 at 14:41 from IEEE Xplore. Restrictions apply.