2 HEYMAN, D.P., TABATABI, A., and LAKSHMAN, T.V.: 'Statistical analysis and simulation study of video teleconference traffic in ATM networks', *IEEE Trans. Circuits Syst. Video Technol.*, 1992, **CSVT-2**, pp. 49–59
3 SEN, P., MAGLARIS, B., RIKLI, N., and ANASTASSIOU, D.: 'Models for packet switching of variable-bit-rate video sources', *IEEE J. Sel. Areas Commun.*, 1989, **SAC-7**, pp. 865–869
4 YEUNG, M.M., and YEO, B.: 'Video visualisation for compact presentation and fast browsing of pictorial content', *IEEE Trans. Circuits Syst. Video Technol.*, 1997, **CSVT-7**, pp. 771–785
5 YEO, B., and YEUNG, M.M.: 'Analysis and synthesis for new digital video applications'. IEEE ICIP Conf. Proc., 1997, Vol. 1, pp. 1–4

Batch verifying multiple DSA-type digital signatures

L. Harn

The digital signature standard proposed by the US government in 1991 is an ElGamal-type signature scheme based on the discrete logarithm problem. Since verifying each ElGamal-type signature requires at least two modular exponentiations, and modular exponentiation is a computationally intensive operation, it becomes desirable to use special purpose hardware or an efficient software algorithm to speed up the signature verification process. The author proposes efficient and secure algorithms to verify multiple digital signatures based on the discrete logarithm. Instead of verifying each individual signature separately, it is proposed to verify multiple signatures simultaneously. The proposed batch verification algorithm can maintain a constant verification time as to verify a single signature.

Introduction: At this time, there are two popular public-key algorithms which can provide digital signatures: the RSA scheme [1], 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 signature scheme [2]; in which the difficulty of breaking the scheme is based on solving the discrete logarithm problem.

In 1991, the US government proposed the digital signature standard (DSS) as a federal standard to enable federal government agencies to use the digital signature algorithm (DSA) [3] to sign electronic documents. The DSS is an ElGamal-type signature scheme based on the discrete logarithm problem. Verifying each ElGamal-type signature requires at least two modular exponentiations. Since modular exponentiation with a very large modulus is a very time-consuming computation, it is desirable to use special purpose hardware or an efficient software algorithm to speed up the signature verification process. Naccache *et al.* [4] proposed an interactive DSA batch verification protocol, in which the signer generates t signatures through interactions with the verifier, and then the verifier validates these t signatures at once based on one batch verification criterion. Lim and Lee [5] pointed out that the interactive DSA batch protocol proposed by Naccache *et al.* is insecure. Later, Harn [6] proposed a DSA-type secure interactive batch verification protocol.

In this Letter, we propose efficient and secure non-interactive algorithms to verify multiple DSA-type digital signatures signed by a private key. Instead of verifying each individual signature separately, we propose to verify multiple signatures simultaneously. Since this approach maintains the same computational load as for verification of a single signature, a significant reduction in time for signature verification can be achieved. The application of our algorithm can be found in some traffic congested gateways that require verification of X. 509 public-key certificates signed by the same certificate authority (CA).

Digital signature algorithm (DSA): The DSA is a signature scheme based on the ElGamal [2] and Schnorr's signature schemes [7]. The DSA parameters are composed by public information p, q, g, a public key y and a secret key x, where

(i) p is a large prime integer of length between 512 and 1024 bits
(ii) q is a 160 bit prime divisor of $p-1$
(iii) g is an element of order q in GF (p)
(iv) x is the secret key of the signer in GF (q)
(v) $y = g^x \bmod p$ is the public key of the signer.

For each message m to be signed, a new random integer k from [1, q–1] is privately selected and then r is computed as $r = (g^k \bmod p) \bmod q$. Instead of signing the message m directly, all ElGamal-type signature schemes should sign the one-way hash result of m. For simplicity, we will ignore the one-way hash function in the following discussion. The DSA signature of m is the pair $\{r, s\}$, where $s = k^{-1}(m + xr) \bmod q$, and $kk^{-1} \bmod q = 1$. A signature $\{r, s\}$ of a message m can be publicly verified by checking that $r = (g^{ms'}y^{rs'} \bmod p) \bmod q$, where $s' = s^{-1} \bmod q$.

Naccache et al. batch verification algorithm: We will use the following example to illustrate the Naccache *et al.* batch verification algorithm. Assume that there are three messages m_1, m_2 and m_3 needing to be signed by the signer. The signer interacts with the verifier and generates three individual signatures $\{r_1, s_1\}$, $\{r_2, s_2\}$, $\{r_3, s_3\}$ of messages m_1, m_2 and m_3, respectively, based on the DSA algorithm. The verifier checks these signatures based on the following batch verification criterion as

$$r_1 r_2 r_3 = (g^{m_1 s'_1 + m_2 s'_2 + m_3 s'_3} y^{r_1 s'_1 + r_2 s'_2 + r_3 s'_3} \bmod p) \bmod q$$

This verification criterion is obtained by multiplying three individual verification equations together.

Lim and Lee's attack: Based on Lim and Lee's attack, any attacker can easily forge signatures to satisfy the batch verification criterion without knowing the secret key. First, the attacker randomly selects $\{u_i, v_i\}$ and computes $r_i = g^{u_i} y^{v_i} \bmod p$, for $i = 1, 2, 3$. The attacker then computes s'_1 that satisfies $v_1 = r_1 s'_1 \bmod q$. Now, the attacker needs to solve s'_2 and s'_3 to satisfy the following two equations:

$$u_1 + u_2 + u_3 = m_1 s'_1 + m_2 s'_2 + m_3 s'_3 \bmod q$$
$$v_1 + v_2 + v_3 = r_1 s'_1 + r_2 s'_2 + r_3 s'_3 \bmod q$$

The problem of the Naccache *et al.* approach is that although the DSA is a secure algorithm to produce individual signatures, the batch verification criterion used to verify multiple signatures is insecure.

Secure DSA-type algorithms: We assume that DSA-type algorithms share the same parameters and similar signing and verification forms as the original DSA algorithm. The following DSA-type digital signature algorithm is based on one of 18 ElGamal-type digital signature schemes developed in [8]. For each message m to be signed, a new random integer k from [1, q–1] is privately selected and then r is computed as $r = (g^k \bmod p) \bmod q$. With knowledge of the secret key x, the signer solves s that satisfies $s = rk - mx \bmod q$. $\{r, s\}$ is the signature of m. With knowledge of the signer's public key y, the signature can be verified by checking whether $r = (g^{sr'}y^{mr'} \bmod p) \bmod q$, where $r' = r^{-1} \bmod q$.

We assume that there are t signatures $\{r_1, s_1\}$, $\{r_2, s_2\}$, ..., $\{r_t, s_t\}$ of t different messages m_1, m_2, ..., m_t, respectively. These t signatures satisfy the following t equations:

$$r_1 = (g^{s_1 r'_1} y^{m_1 r'_1} \bmod p) \bmod q$$
$$r_2 = (g^{s_2 r'_2} y^{m_2 r'_2} \bmod p) \bmod q$$
$$\vdots$$
$$r_t = (g^{s_t r'_t} y^{m_t r'_t} \bmod p) \bmod q$$

By multiplying these t equations together, we obtain the batch verification criterion as

$$r_1 r_2 \ldots r_t =$$
$$(g^{s_1 r'_1 + s_2 r'_2 + \ldots + s_t r'_t} y^{m_1 r'_1 + m_2 r'_2 + \ldots + m_t r'_t} \bmod p) \bmod q$$

We claim that the verifier can verify these t signatures simultaneously by checking this batch verification criterion. If both sides of the equation are identical, the t signatures $\{r_1, s_1\}$, $\{r_2, s_2\}$, ..., $\{r_t, s_t\}$ of messages m_1, m_2, ..., m_t can be verified. However, if the identity does not hold, where this may result from either transmission noise or invalid individual signatures, we need to verify each individual signature separately. It is obvious that to verify these t signatures according to the batch verification criterion requires 2 modular exponentiations. However, if the verifier verifies each

individual signature separately, it requires $2t$ modular exponentiations.

Security and discussion: We would like to point out that Lim and Lee's attack does not work properly in this proposed algorithm. In their attack, the attacker can randomly select all r_i first. Then the attacker can solve s_i accordingly. However, in our proposed algorithm, r_i cannot be randomly selected in the first place. Conversely, the signature algorithm used to sign each individual signature is also secure. In fact, the following ElGamal-type signature algorithms as listed in [8] can also be used to develop similar batch verification criteria.

signature equation	signature verification
$rx = mk + s \bmod p-1$	$y^r = r^m\alpha^s \bmod p$
$sx = rk + m \bmod p-1$	$y^s = r^r\alpha^m \bmod p$
$sx = mk + r \bmod p-1$	$y^s = r^m\alpha^r \bmod p$
$rmx = k + s \bmod p-1$	$y^{rm} = r\alpha^s \bmod p$
$x = mrk + s \bmod p-1$	$y = r^{mr}\alpha^s \bmod p$
$sx = k + mr \bmod p-1$	$y^s = r\alpha^{mr} \bmod p$
$sx = rmk + l \bmod p-1$	$y^s = r^{rm}\alpha \bmod p$
$(r + m)\, x = k + s \bmod p-1$	$y^{r+m} = r\alpha^s \bmod p$
$x = (m + r)\, k + s \bmod p-1$	$y = r^{m+r}\alpha^s \bmod p$
$sx = k + (m + r) \bmod p-1$	$y^s = r\alpha^{m+r} \bmod p$
$sx = (r + m)\, k + 1 \bmod p-1$	$y^s = r^{r+m}\alpha \bmod p$

Signing and verifying each DSA signature, in addition to computing modular exponentiation(s), requires computation of modular inverses k^{-1} and s^{-1}. The DSA variant proposed in the beginning of the previous section makes computation easier for the signer by not requiring him to compute $k^{-1} \bmod q$. However, the verifier still needs to compute $r^{-1} \bmod q$. Note the following two ElGamal-type signature schemes that make computation easier on both the signer and the verifier by not forcing them to compute either k^{-1} or r^{-1}.

signature equation	signature verification
$rmx = k - s \bmod p-1$	$r = \alpha^s y^{rm} \bmod p$
$(r + m)\, x = k - s \bmod p-1$	$r = \alpha^s y^{r+m} \bmod p$

We claim that these two variants are the most efficient DSA variants since they are as secure as the DSA, they do not require computation of any modular inverse on the signer and the verifier, and they can support batch verifying multiple signatures.

Conclusion: We have proposed efficient and secure algorithms to verify multiple digital signatures. Instead of verifying each individual signature separately, we propose to verify multiple signatures simultaneously. Our algorithm can almost maintain a constant time for verifying multiple signatures.

4 March 1998
Electronics Letters Online No: 19980620

L. Harn (*Department of Computer Networking, University of Missouri-Kansas City, MO 64110, USA*)

References

1 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
2 ELGAMAL, T.: 'A public-key cryptosystem and a signature scheme based on discrete logarithms', *IEEE Trans. Inform. Theory*, 1985, **IT-31**, pp. 469–472
3 : 'Proposed Federal Information Processing Standard for Digital Signature Standard (DSS)', in *Federal Register*, 1991, **56**, (169), pp. 42980–42982
4 NACCACHE, D., M'RAIHI, D., RAPHEALI, D., and VAUDENAY, S.: 'Can DSA be improved: complexity trade-offs with the digital signature standard'. Pre-Proceedings of Eurocrypt '94, pp. 85–94
5 LIM, C.H., and LEE, P.J.: 'Security of interactive DSA batch verification', *Electron. Lett.*, 1994, **30**, (19), pp. 1592–1593
6 HARN, L.: 'DSA type secure interactive batch verification protocol', *Electron. Lett.*, 1995, **31**, (4), pp. 257–258
7 SCHNORR, C.P.: 'Efficient signature generation by smart cards', *J. Cryptol.*, 1991, **3**, (3), pp. 161–174
8 HARN, L., and XU, Y.: 'Design of generalised ElGamal type digital signature schemes based on discrete logarithm', *Electron. Lett.*, 1994, **30**, (24), pp. 2025–2026

MacDES: MAC algorithm based on DES

L.R. Knudsen and B. Preneel

A new message authentication code (MAC) algorithm is proposed, which improves the popular retail MAC based on the data encryption standard; it has the same complexity, but provides better resistance against key recovery attacks. In addition, a new key recovery attack on the retail MAC is presented, requiring a single known text-MAC pair and 2^{56} online MAC verifications.

Introduction: Message authentication code (MAC) algorithms are widely used to provide data integrity and data origin authentication. They are preferred over digital signatures for applications with inexpensive processors, such as smart cards, or high data rates, such as IP level security. MACs are used in settings where sender and receiver share a secret key K, of bitlength k. The sender sends, with the message x, an $m$bit string $MAC_K(x)$, that is a complex and non-invertible function of every bit of K and x. Typically m is between 32 and 64 bit. The receiver can verify that the message x does indeed come from the claimed sender by recomputing the value $MAC_K(x)$, and verifying it against the transmitted MAC value. An active eavesdropper can modify the message, but faces the task of determining the MAC value without knowing K.

The main security property for a MAC is that it should be resistant to forgery, i.e. it must be computationally infeasible for someone who does not know the secret key to find an arbitrary new message x and the corresponding value $MAC_K(x)$. One strategy is to guess the MAC value; it has success probability $1/2^m$, which means that it can be easily defeated by choosing m large enough. A more advanced strategy consists in asking the MAC value for a number of chosen texts, or verifying a number of text-MAC values before coming up with a forgery that has a high probability of being correct.

A second attack on a MAC is a key recovery attack; recovering the key allows for an arbitrary forgery. One approach is to search the key space exhaustively. This requires about k/m text-MAC pairs (to define the key uniquely), and 2^{k-1} MAC evaluations. It can be precluded by choosing k appropriately; 80–90 bit should be sufficient for long term security.

CBC-MAC: The standard MAC algorithm for banking applications is CBC-MAC [1 – 3] based on the data encryption standard (DES) [4]. The block length n and key length k of CBC-MAC are equal to the block and key length of the block cipher on which it is based ($n = 64$ and $k = 56$ for DES). The input is padded unambiguously to a multiple of the block length, and then divided into t blocks x_1 through x_t. The following iteration is performed:

$$H_i = E_K(H_{i-1} \oplus x_i) \qquad 1 \le i \le t$$

Here $E_K(x)$ denotes the encryption of x using key K with an $n$bit block cipher E and $H_0 = 0$. The MAC value is then computed as $MAC_K(x) = g(H_t)$, where g is the output transformation. The mapping g is intended to preclude a simple forgery attack (see e.g. [5]). One approach is for g to select the leftmost m bits, but it has been shown in [6] that this is less secure than expected.

The ANSI retail MAC [1] selects as its output transformation a decryption with a second key K_2 followed by an encryption with K_1 (such that the last block undergoes a two key triple encryption):

$$g(H_t) = E_{K_1}(D_{K_2}(H_t)) = E_{K_1}(D_{K_2}(E_{K_1}(x_t \oplus H_{t-1})))$$

Here D denotes decryption. This alternative is widely used because it requires very little overhead (two encryptions), and has the additional advantage of precluding an exhaustive search against the 56 bit DES key [7]. With a 112 bit key, one can expect that the retail MAC based on DES is resistant to key recovery attacks.