February 2002

DSA

Security Evaluation of the Signature Scheme and Primitive

Evaluator: Prof. Jean-Jacques Quisquater, Math RiZK, consulting

Scientific Support: Dr. François Koeune, K2Crypt

Executive summary.
This report is a security evaluation of the standard signature algorithm DSA. The main conclusion is that the implementers need to be very careful and trusted. The length of the main parameter (1024 bits) needs to be evaluated correctly: it means that very soon the value of about 1500 bits will be a useful addition and the relevant 160 bit parameters must be ajusted to 256 bits for taking into account a level of security valid for the next 20 years.

Contents

1 General description

1.1 DSA parameters

The signature algorithm `DSA` makes use of the following parameters:

p: a prime number;

q: a prime number, divisor of $p-1$;

g: an element of order $q \bmod p$;

x: a randomly chosen integer with $0 < x < q$, associated to a given signer;

k: a randomly chosen integer with $0 < k < q$, only valid for one signature.

The integers p, q and g are the system parameters and can be public and used by many signers. The private signing parameter of a given signer is x and the corresponding public verification parameter of the same signer is the value $y = g^x \bmod p$.

1.2 Signature generation

The signature σ of the message m is the pair of numbers r and s computed according to the following formulas, where `SHA` is the Standard Hash Algorithm (described in the FIPS 180 from NIST):

$$r = (g^k \bmod p) \bmod q,$$

$$s = (k^{-1} \cdot (\texttt{SHA}(m) + x \cdot r)) \bmod q.$$

1.3 Signature verification

The inputs consist of:

— y: the public verification parameter of the signer;

— p, q, g: the system parameters;

— $\texttt{SHA}(m)$: the hashing of m;

— r, s: the signature σ.

The verifier first checks to see that $0 < r < q$ and $0 < s < q$; if either condition is violated the signature shall be rejected. He then computes

$$((g^{\mathtt{SHA}(m)\cdot s^{-1} \bmod q}) \cdot y^{r\cdot s^{-1} \bmod q}) \bmod p) \bmod q$$

and compares to r. If the equality occurs the verifier accepts the message and the signature: otherwise the verifier rejects the message and the signature.

1.4 Parameters size

The standard specifies $2^{159} < q < 2^{160}$ and $2^{L-1} < p < 2^{L}$ for $512 \leq L \leq 1024$ and L a multiple of 64 (however, FIPS 186-2 rev. 1 forces $L = 1024$: see section 2). Additionally, this report will show that k must be a 160-bit random number (see section 4.1.2).

1.5 Parameters generation

The standard also specifies algorithms for parameters generation, in particular for random numbers generation. The use of these techniques is not mandatory, provided another FIPS-approved method is used instead.

As will be showed in following sections, this aspect of the standard must not be neglected, as several attacks are based on the assumption that other generation techniques are used.

For the sake of conciseness, we will not repeat this generation procedure here. We refer the reader to the standard (FIPS 186-2, appendices 2, 3 and 4 + change notice 1) for full details.

2 Evolution of the standard

FIPS 186, the first version of the Digital Signature Standard (DSS) was issued in May 1994. In May 1997, the National Institute of Standards and Technology announced a revision to allow the use of `RSA` or elliptic curves `DSA` as alternatives to the `DSA` . A first revision (FIPS 186-1) issued in December 1998, added `RSA` as specified in ANSI X9.31. In January 2000, a second revision (FIPS 186-2), addressed the `ECDSA` question (as specified in ANSI X9.62).

The current version of the standard is FIPS 186-2, but, in August 2001, a change notice was issued by the NIST. This change notice, which took effect in October 2001, forces the modulus size (formerly between 512 and 1024 bits) to

1024 bits and corrects a flaw of the random generator (this point is described in more details below).

3 Security arguments

The security of the `DSA` relies on two distinct but related discrete logarithm problems. One is the logarithm problem in $\mathbb{Z}_p^*$ where the powerful index-calculus methods apply; the other is the logarithm problem in the cyclic subgroup of order q, where the best current methods run in "square-root" time (we refer the reader to [19] for more details on discrete logarithm calculation methods).

3.1 Security proof

A slight variation of `DSA` , in which $\texttt{SHA}(m)$ is replaced by $\texttt{SHA}(m, r)$, has been proved by Brickell et al. [3] to be secure in the random oracle model.

More precisely, it has been proved that, if `DSA` can be broken by an existential forgery using an adaptatively chosen-message attack, then either:

- the discrete logarithm problem can be solved, or
- `SHA` can be distinguished from an ideal hash function, or
- multi-collisions for the function "$x \mapsto (g^x \bmod p) \bmod q$" can be found.

The first case corresponds to a well-known problem, which is widely believed by the scientific community to be hard to solve; the second would imply a serious failure of `SHA` , requiring its withdrawal as a hash function; the third would highlight a serious unwanted weakness resulting from the "projection" from $g^x \bmod p$ to $(g^x \bmod p) \bmod q$.

We refer the reader to [3] for a more formal definition of the security proof.

3.2 Importance of k's secrecy

It is very important for the random number k to be kept secret by the signer. As a matter of fact, if k was known, it would suffice for the attacker to compute

$$(s \cdot k - \texttt{SHA}(m)) \cdot r^{-1} \bmod q$$

to immediately obtain the signer's secret key. Similarly, a different k (nonce) must be selected for each message signed; otherwise, the private key can be determined with high probability as follows. Suppose

$$s_1 = k^{-1}(\mathtt{SHA}(m_1) + x \cdot r) \bmod q$$
$$s_2 = k^{-1}(\mathtt{SHA}(m_2) + x \cdot r) \bmod q.$$

Then[1]

$$k = (s_1 - s_2)^{-1}(\mathtt{SHA}(m_1) - \mathtt{SHA}(m_2)) \bmod q.$$

Once k is known, the above attack can be applied.

Several attacks (that will be described in next sections) against DSA are based on (partial) knowledge of the nonce k.

4 Overview of currently known attacks

4.1 General-purpose attacks

4.1.1 Random generation of k

It is a well-known fact that linear congruential generators (LCGs) are not cryptographically secure: by observing a part of the output, it is possible to predict all generator's future output [8, 9, 11, 16]. Due to their ease of implementation and good performances, one may however feel tempted to use LCGs to produce not-so-sensitive data, such as the nonces k required for each individual signature. This may be encouraged by the fact that, as the output of the generator is never revealed to the adversary, the aforementioned attacks do not apply. This, however, is simply not true.

In [2], Bellare, Goldwasser and Micciancio show, using using lattice reduction-based techniques ([17, 1]), how the secret key can be quickly recovered after seeing a few DSA signatures.

Countermeasures The countermeasure against this attack is very simple: do not use linear congruential generators or truncated linear congruential generators to generate RSA parameters, even for nonces. In particular, we point out that this attack does not apply against a strict application of the standard, since this requires the use of FIPS-approved random generators.

[1] This supposes that $(s_1 - s_2) \neq 0 \bmod q$, which is satisfied with high probability.

4.1.2 Partial knowledge of k

In [21, 22] (which improve a similar result in [10]), Nguyen and Shparlinski present a polynomial-time algorithm that recovers the DSA secret key when a few consecutive bits of the nonces k for a number of signatures at most linear in $\log(q)$.

The attack, based on lattice reduction techniques, is too complex to be described in this report (we refer the interested reader to [21]). Its efficiency is however impressive: according to the authors, practical experiments based on the observation of 70 DSA signatures (with a 512-bit prime p) allowed to recover the secret key in 100% of the trials if the 5 least significant bits of each nonce are known, and in 90% of the trials if 4 bits are known. Practical experiments were unsuccessful with fewer bits revealed, but the authors insist on the fact that this does not mean the attack does not apply in this case (they conjecture that 2 bits should be feasible in practice, and even 1 single bit could perhaps be reached). The attack also applies, with same efficiency, if the *most* significant bits of the nonces are known. A similar attack, but requiring twice as many bits, applies to an arbitrary portion of consecutive bits of the nonces.

It is worth noting that if, for efficiency reasons, one chooses nonces k with fewer bits than q, then this attack obviously applies.

Countermeasures This result is yet another witness of the fact that the secrecy of k is fundamental for the security of the scheme. k *must* be a 160-bit value generated by a cryptographically secure random generator. Note that this does in no way mean that the first bit of this 160-bit value has to be *forced* to 1 (on the contrary, this would be an open door to the above attack).

4.1.3 Subliminal channel

Simmons [28, 27] showed the existence of a subliminal channel in the DSA , which allows messages to be secretly embedded into signatures in such a way that only the intended receiver of the message will be able to notice its presence and recover it. Other users will simply see a signature that they can verify.

A non negligible consequence of this subliminal channel is that it would allow an unscrupulous implementer to leak a part of the user's private key with each produced signature. The lesson of this is that one should never use a DSA implementation if he does not trust the implementer.

4.1.4 Hidden collisions

Vaudenay [29] showed that it is possible for a dishonest authority to forge DSA public parameters in such a way that two – previously chosen – messages will always produce the same signatures (what they call a collision), independently of the secret key used.

A possible exploit of this weakness is the following. The attacker chooses two messages, one anodine and one of great interest for him (a bank transfer order, for example), and generates the public parameters that will produce a collision for these messages. He then convinces the user to sign the first message, and hands the result as a signature of the second.

Description of the attack As Vaudenay points out, the actual hash function involved in DSA is not the $\texttt{SHA}(\cdot)$ function, but rather $\texttt{SHA}(\cdot) \bmod q$. The collision problem for DSA is therefore to find a pair of messages (m, m') such that

$$\texttt{SHA}(m) \equiv \texttt{SHA}(m') \pmod q.$$

Since q is 160-bit long, this problem is still infeasible. However, we have one more degree of freedom, in the form of the parameter q.

Concretely, from a random pair (m, m'), we check whether or not $q = |\texttt{SHA}(m) - \texttt{SHA}(m')|$ is a 160-bit prime. From the Prime Number Theorem, we deduce that, with an average of 222 trials, we obtain a collision which defines a valid prime q. It is then not difficult to derive valid p and g from this q.

Countermeasures As we have seen, the attack requires the generation of a special-form parameter q. The attack can therefore not be carried out if the parameter generation procedure recommended by the standard is followed. Remember that this procedure makes it possible – and even mandatory – to prove that the procedure has been correctly followed, by providing the user with the initial SEED value (certificate of good forgery). It is therefore fundamental that the user checks this certificate of good forgery before accepting public parameters.

Vaudenay proposed a variant of his attack working even if this generation procedure is followed, but its complexity 2^{74}, although better than the birthday attack, makes it unrealistic. He also proposed a change in the generation algorithm that would counter his attack (we mean, the variant aimed at FIPS generation procedure), but this change has not been applied in subsequent versions of the standard.

4.1.5 FIPS 186 RNG flaw

As was said before, FIPS 186 specifies random generation methods for parameters x (appendix 3.1) and k (appendix 3.2).

On November 15, 2000, at a meeting of the IEEE P1363 working group, Daniel Bleichenbacher presented an attack exploiting a weakness in DSA 's random number generator. According to the meeting minutes [7]: *"The results presented were preliminary and Bleichenbacher requested that the group not publicly disclose the contents of the presentation until after February 15th. [...] A reference to this attack will be included [in P1363's resulting security note] when there is a paper available to reference."*

As a consequence of this attack, the NIST released a revision proposal of the standard (FIPS 186-2: revision 1), fixing this weakness.

Bleichenbacher's attack has not been published since then, but, in view of the (imprecise) information available from press releases, and, above all, of the changes proposed for the subsequent version of the standard, we conclude that the exploited weakness was the following.

In the two generation algorithms (more precisely, at step 3.c of algorithm 3.1, and at step 3.a of algorithm 3.2), a temporary value is processed through a one-way function $G(t, c)$, and the result is reduced modulo q. Several methods are proposed to construct this one-way function, but the point of interest is that its output is 160-bit long. The output range of this function is thus of size 2^{160}.

On the other hand, since q is a 160-bit number, the output range of the reduction $\bmod q$ is smaller than 2^{160}. As a consequence, collisions will occur during this reduction.

More precisely, values in the interval $[0, 2^{160}-1-q]$ will be twice more probable (probability 2^{-159}) than values in the interval $[2^{160}-q, q-1]$ (probability 2^{-160}), whereas one would expect uniform distribution (probability $1/q$).

NIST's proposed random number generator is thus biased. It is this bias that Bleichenbacher exploits.

Very little information is available regarding the attack itself, but, according to FIPS's change notice, the attack has a workfactor of 2^{64} and requires 2^{22} known signatures. This workfactor is still out of reach for today's computing power. Consequently, although we strongly recommend to implement the recommended change in the RNG and to limit current systems to no more than 2 millions signatures with a specific key pair, we do not believe that this attack – at least, in its current version – represents a significant threat for current systems.

The fix proposed in the standard's revision solves this problem by concatenating two consecutive outputs of the G function into a 320-bit number before reducing it modulo q. Although this does not completely remove the bias, it nevertheless reduces it to something negligible.

4.2 Implementation attacks

4.2.1 OpenPGP weakness

In 1998, the format of OpenPGP messages was published as a RFC [4]. The goal of this document was to publish all necessary information so that various interoperable applications could be created on that basis.

In 2001, Klima and Rosa [12] published an attack exploiting insufficient integrity protection in OpenPGP format. This attack, that could possibly affect all OpenPGP-compatible applications[2] is capable of recovering `DSA` secret parameters by tampering with the public parameters and obtaining one single signature based on the corrupted values. A similar attack can be applied against `RSA` .

Description of the attack The attack basically goes as follows. The initial weakness which is exploited is the fact that, in the "Secret Key Packet" data structure of OpenPGP, only the secret parameter x is stored in an encrypted way (and protected by a checksum); public parameters p, q, g and y are simply stored in clear, and can therefore be tampered with if the attacker has access to the secret key file. As the fact that "sensitive" data are encrypted may give the user a feeling of confidence, gaining access to this file may not be unrealistic for an attacker, for example during a transfer on a floppy disk.

Now, the attacker replaces p by p' such that

1. p' is 159-bit long and $p' < q$;
2. p' has the form $p' = t * 2^s + 1$, with s big and t a small prime number.

The authors of the attack suggested the value (in hexadecimal notation) $p' = 0x538000000000000000000000000000000000000001$, which corresponds to $s = 151$ and $t = 167$.

The attacker also replaces g by g' such that

[2]Although correct integrity checking on the parameters before processing them defeats the attack, so applications must be reviewed on a case-by-case basis to determine whether they are immune to this attack or not.

1. $1 < g' < p' - 1$
2. g' is a generator of the multiplicative group $\mathbb{Z}^*_{p'}$.

The next time the user will generate a signature, he will unknowingly produce, instead of the usual (r, s) pair, a pair (r', s') such that

1. $r' = ((g')^k \bmod p') \bmod q$, for some secret value k;
2. $s' = ((k^{-1} \bmod q)(h(m) + xr')) \bmod q$.

Since $p' < q$, the first relationship is equivalent to

$$r' = (g')^k \bmod p'.$$

The key issue is that the discrete log problem has been transferred into a group in which it is easy to solve, due to the special form of p'. It is therefore easy for the attacker to recover k from this signature. As we have seen before, it is then obvious to deduce the secret key x from k. The detailed algorithm for computing the discrete log in $\mathbb{Z}^*_{p'}$ can be found in [12].

Countermeasures Until an adjustment of the OpenPGP format occurs, Klima and Rosa suggest implementing the following integrity checks before a `DSA` signature is performed:

1. $p, q, g, x, y > 0$
2. p is odd, q is odd
3. $2^{159} < q < 2^{160}$
4. $1 < g < p$
5. $1 < y < p$
6. $x < q$
7. $q | (p - 1)$
8. $g^q \bmod p = 1$
9. $g^x \bmod p = y$

4.2.2 Cryptolib potential weakness

Bleichenbacher noticed (according to a private communication in [22]) that in AT&T's Cryptolib version 1.2, the implementation of DSA suffers from the following flaw: the random nonce k is always odd, thus leaking its least significant bit (probably because the same implementation is used for El Gamal). In current state of the art, no immediate attack can actually exploit this weakness, but, in light of the attack explained in section 4.1.2, this could be a serious weakness.

4.2.3 Timing attack

The modular exponentiation $r = (g^k mod p) mod q$ can most probably be subject to a timing attack [13]. In particular, timing attacks developed against RSA [5, 25, 26] should be immediately applicable to the DSA .

Power analysis (SPA, DPA [15, 14]) and electromagnetic analysis (SEMA, DEMA [23, 24]) should be applicable as well. However, we point out that this is the case for most cryptographic systems, and therefore does not constitute a real weakness of DSA . Physical attacks target specific implementations rather than cryptosystems in general. Therefore, protecting against such attacks is an implementation issue, that must be handled depending on the destination platform, and its potentiality to be subject to such an attack.

4.3 Other remarks

4.3.1 Timestamping

The Fact sheet on Digital signature standard[3] states that: *"In legal systems, it is often necessary to affix a time stamp to a document in order to indicate the date and time at which the document was executed or became effective. An electronic time stamp could be affixed to documents in electronic form and then signed using the DSA. Applying the DSA to the document would protect and verify the integrity of the document and its time stamp."*

We however insist on the fact that the DSA does not provide a secure time stamping capability, since the signer could easily forge any time stamp for his own documents.

[3]See http://www.nist.gov/public_affairs/releases/digsigst.htm

4.3.2 Batch verification

In the preprint version of [20], Naccache et al. proposed several methods to optimize DSA's performances. Among them are some batch verification methods, which allow to quickly verify a set of `DSA` signatures.

One of the proposed methods was proved by Lim and Lee [18] to be completely insecure. However, it seems this method was finally removed from the final proceedings version of [20].

5 Summary

Concerning the security of 1024 bits DSA, I really think it is marginal for 10 years. I mean: it will be dangerous for the most secure applications. See the table *http://www.cryptosavvy.com/table.htm* where you see that they propose a length of 1464 bits to be secure till 2012. I don't agree completely with this table and I'll do again their computations analysing their hypothesis. They also use software with classical computers not the faster and more adequate circuits FPGA for cryptanalysis (see, for instance, the fuss with the recent proposal by Dan Berstein about a possible fast implementation of factorization and logarithm computations). Some progress on cryptanalytic algorithms are also possible.

Finally, several points raised in this report need to be clearly stated to the management of any team intending to use DSA. These include:

- do not use a `DSA` implementation if you do not trust the implementer;
- actually implement integrity tests,
- check certificate of good forgery,
- ...

References

[1] L. Babai, *On lovàsz' lattice reduction and the nearest lattice point problem*, Combinatorica **6** (1986), no. 1, 1–13.

[2] M. Bellare, S. Goldwasser, and D. Micciancio, *"pseudo-random" number generation within cryptographic algorithms: the DSS case*, Advances in Cryptology - CRYPTO '97, Santa Barbara, California (B.S. Kaliski Jr., ed.), LNCS, vol. 1294, Springer, 1997, pp. 277–291.

[3] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung, *Design validations for discrete logarithm based signature schemes*, Proc. of PKC '2000 (Berlin), Springer-Verlag, 2000, Lecture Notes in Computer Science Volume 1751, pp. 276–292.

[4] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, *RFC 2440: OpenPGP message format*, November 1998.

[5] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, *A practical implementation of the timing attack*, Proc. CARDIS 1998, Smart Card Research and Advanced Applications (J.-J. Quisquater and B. Schneier, eds.), LNCS, Springer, 1998.

[6] E. El Mahassni, P.Q. Nguyen, and I.E. Shparlinski, *The insecurity of Nyberg-Rueppel and other DSA-like signature schemes with partially known nonces*, Proc. of Workshop on Lattices and Cryptography (Berlin), Springer-Verlag, 2001, Lecture Notes in Computer Science Volume 1109.

[7] IEEE P1363 Working Group for Public-Key Cryptography Standards, *Meeting minutes of wednesday, november 15, 2000 - NSA, Fort Meade, MD*, available from `http://grouper.ieee.org/groups/1363/WorkingGroup/minutes/Nov00.txt`, 2000.

[8] A.M. Frieze, R. Kannan, and J.C. Lagarias, *Linear congruential generators do not produce random sequences*, Proc. of 25th IEEE Symposium on Foundations of Computer Science, IEEE, 1984, pp. 480–484.

[9] J. Hastad and A. Shamir, *The cryptographic security of truncated linearly related variables*, Proc. of 17th ACM Symposium on Theory of Computing, ACM, 1985, pp. 356–362.

[10] N.A. Howgrave-Graham and N.P. Smart, *Lattice attacks on digital signature schemes*, Desing, codes and cryptography, 2001, to appear.

[11] A. Joux and J. Stern, *Lattice reduction: a toolbox for the cryptanalyst*, Journal of cryptology **11** (1998), no. 3, 161–185.

[12] V. Klima and T. Rosa, *Attack on private signature keys of the OpenPGP format, PGP programs and other applications compatible with OpenPGP*, available from `http://www.i.cz/en/pdf/openPGP_attack_ENGvktr.pdf`, 2001.

[13] P. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology - CRYPTO '96, Santa Barbara, California (N. Koblitz, ed.), LNCS, vol. 1109, Springer, 1996, pp. 104–113.

[14] P. Kocher, Jaffe J., and B. Jub, *Differential power analysis*, Proc. of Advances in Cryptology – CRYPTO '99 (M. Wiener, ed.), LNCS, vol. 1666, Springer-Verlag, 1999, pp. 388–397.

[15] P. Kocher, J. Jaffe, and B. Jun, *Introduction to differential power analysis and related attacks*, `http://www.cryptography.com/dpa/`, 1998.

[16] H. Krawczyk, *How to predict congruential generators*, Advances in Cryptology - Crypto '89 (Berlin) (Gilles Brassard, ed.), Springer-Verlag, 1989, Lecture Notes in Computer Science Volume 435, pp. 138–153.

[17] A.K. Lenstra, H.W. Lenstra, and L. Lovàsz, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982).

[18] C.H. Lim and P.J. Lee, *Security of interactive DSA batch verification*, Electronics letters, vol. 30, 1994, pp. 1592–1593.

[19] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[20] D. Naccache, D. M'raihi, S. Vaudenay, and D. Raphaeli, *Can D. S. A. be improved? complexity trade-offs with the digital signature standard*, Advances in Cryptology - EuroCrypt '94 (Berlin) (Alfredo De Santis, ed.), Springer-Verlag, 1995, Lecture Notes in Computer Science Volume 950, pp. 77–85.

[21] P.Q. Nguyen, *The dark side of the hidden number problem: Lattice attacks on DSA*, Proc. of Workshop on Cryptography and Computational Number Theory (CCNT'99) (K.-Y. Lam, I. E. Shparlinski, H. Wang, and Xing C., eds.), 2001, pp. 321–330.

[22] P.Q. Nguyen and I.E. Shparlinski, *The insecurity of the Digital Signature Algorithm with partially known nonces*, Journal of cryptology, to appear.

[23] Jean-Jacques Quisquater and David Samyde, *A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods*, Eurocrypt rump session, 2000.

[24] ______, *Electromagnetic analysis (EMA): measures and countermeasures for smart cards*, Smart cards programming and security (e-Smart 2001), Lectures Notes in Computer Science (LNCS), vol. 2140, Springer, 2001, pp. 200–210.

[25] W. Schindler, *Optimized timing attacks against public key cryptosystems*, Statistics & Decisions (2000), to appear.

[26] W Schindler, F. Koeune, and J.-J. Quisquater, *Unleashing the full power of timing attack*, submitted, 2001.

[27] G. J. Simmons, *The subliminal channel in the U.S. Digital Signature Algorithm (DSA)*, Proc. of 3rd Symposium on State and Progress of Research in Cryptography - SPRC'93, 1993, pp. 35–54.

[28] ______, *Subliminal communication is easy using the DSA*, Advances in Cryptology - EuroCrypt '93 (Berlin) (Tor Helleseth, ed.), Springer-Verlag, 1993, Lecture Notes in Computer Science Volume 765, pp. 218–232.

[29] S. Vaudenay, *Hidden collisions on DSS*, Advances in Cryptology - Crypto '96 (Berlin) (Neal Koblitz, ed.), Springer-Verlag, 1996, Lecture Notes in Computer Science Volume 1109, pp. 83–88.