Minding your p's and q's

Ross Anderson[1], Serge Vaudenay[2]

[1] Computer Laboratory, Pembroke Street, Cambridge, CB2 3QG, England
[2] Ecole Normal Supérieure — DMI, 45 rue d'Ulm, 75230 Paris, France

Abstract. Over the last year or two, a large number of attacks have been found by the authors and others on protocols based on the discrete logarithm problem, such as ElGamal signature and Diffie Hellman key exchange. These attacks depend on causing variables to assume values whose discrete logarithms can be calculated, whether by forcing a protocol exchange into a smooth subgroup or by choosing degenerate values directly. We survey these attacks and discuss how to build systems that are robust against them. In the process we elucidate a number of the design decisions behind the US Digital Signature Standard.

1 Introduction

The large majority of actual attacks on conventional cryptosystems are due to blunders in design, implementation and operation, which are typically discovered by chance and exploited in an opportunistic way [2, 3]. We have not yet accumulated comparable experience of the failure modes of public key cryptosystems, but the early indications are that nothing much has changed: an implementation of the Fiat-Shamir identification scheme [14] is reported to have failed because the application always used the same random challenge [22, 23].

Any system can be compromised by a gross blunder of this kind; but public key systems present a large number of subtle failure modes [5]. Some are variants on the middleperson attack or exploit the semantics of public key systems in general, while others depend on the mathematical features of particular systems.

In this paper, we investigate protocols based on the discrete logarithm problem such as ElGamal signature, Diffie-Hellman key exchange and the Chor-Rivest system. We describe a family of simple attacks, some of them new, which have been found by the authors and others over the last year or two. These attacks enable us to explain the design of the US digital signature standard [24].

2 ElGamal and related protocols

The original ElGamal signature uses a large prime p as its modulus and a generator g of the group Z_p^* as its base. These values are common, and may be issued by authority; each user has a secret key x and a corresponding public key $y = g^x \bmod p$. To sign a message M, the signer first picks a random k which is coprime to $p - 1$ and issues the signature (r, s) where

$$r = g^k \bmod p \tag{1}$$
$$s = \frac{M - xr}{k} \bmod p - 1 \tag{2}$$

Writing the second equation as $sk + xr \equiv M \bmod p - 1$, we see that a valid signature is checked by verifying

$$r^s y^r \equiv g^M \pmod{p}$$

We shall show that this system is quite insecure, unless considerable care is taken in the choice of parameters.

2.1 The effect of the choice of p

Suppose that $p-1 = qw$, where w is smooth (i.e., all its prime factors are small). This was the case with the primes p and q originally proposed with the US digital signature standard (DSS):

$$\frac{p-1}{q} = 2^{70}3^{46}5^{30}7^{25}11^{20}$$

(see [1]). Then the group Z_p^* in which we are operating has a subgroup of order q, in which discrete log is hard, and a subgroup of order w in which it is easy.

The effect is that keys can be recovered modulo w. An attacker can solve the equation $(g^q)^z \equiv y^q \pmod{p}$ in the group of order w, giving x modulo w, and then derive every message key k modulo w from equation (2). This was originally pointed out by van Oorschot and Wiener [32]; and also by Anderson, Vaudenay, Preneel and Nyberg in the context of subliminal channels, where the effect is to create one or more broadband covert channels [6]. However, there are serious practical security implications as well.

In order to generate the message keys k for successive signatures, a random number generator is required, and the use of ElGamal signatures with the above prime p has the effect of disclosing 352 out of every 512 bits of the generator's output. The typical generator has resonances and other features that can be exploited (see for example [20]), with the result that further bits of message keys k might be inferred by an adversary. Once a single key k modulo q is compromised, the signing key x can easily be calculated.

The amount of information leaked will depend on the choice of p, and in particular on the factors of $p-1$. The above choice of p may be the worst possible, but even the best possible — say $p = 2qr + 1$ where both q and r are prime — will still leak one bit of message key per signature. So the choice of p is not enough to completely prevent the leakage of key material. A better approach is to use a base element g that generates a prime order subgroup of Z_p^*; then we are doing our cryptographic operations in a single group, rather than in two groups at a time. This is exactly the approach taken by the designers of DSS, which has

$$r = (g^k \bmod p) \bmod q$$
$$s = \frac{h(M) + xr}{k} \bmod q$$

This blocks the above attack, and also (contrary to the popular misconception) reduces the bandwidth available for use as a subliminal channel.

2.2 Attacks based on interaction between k and g

After the DSS was introduced, there was some controversy about whether a trapdoor could be hidden in the primes p and q by, for example, choosing them to be numbers which could be respresented compactly by a polynomial of degree four or five, so that the authority could calculate discrete logarithms using the number field sieve [11].

But no-one seems to have stopped to think about the choice of base g.

A simple forgery attack on ElGamal consists of chosing $k = (p-1)/2$. Then $r = p-1$, so the secret key x is hidden by r in the product xr modulo $p-1$. Unfortunately, this cannot work unless $M = (p-1)/2$ or $M = p-1$ since k is not invertible modulo $p-1$. But the authority can use this idea to provide valid parameters with a hidden trapdoor.

If the authority chooses the base g such that he knows the discrete logarithm k of $r = (p-1)/2$ (for instance by chosing $g = r^{1/k} \bmod p$ for random invertible k until it is a generator), then he is able to choose this k which enables to compute s (he needs to know $x \bmod 2$ but this can easily be obtained by checking whether y is a quadratic residue or not.)

More generally, the authority can isolate a smooth factor w of $p-1$ and choose g such that he knows the discrete logarithm k of some $j(p-1)/w$ between 0 and $p-1$. The ability to compute s from k is then equivalent to the knowledge of $x \bmod w$ which, as we have seen, is easy when w is smooth.

A similar forgery attack was recently found by Bleichenbacher. Suppose that $p-1 = qw$ as before with w smooth, and that the authority can choose a g which divides w. Then a valid signature can be forged on any message M as

$$r = (p-1)/g \pmod p$$
$$s = (M - ry_q)(p-3)/2 \pmod{p-1}$$

where y_q is the smooth part of the public key y (i.e. a solution of $(g^q)^z \equiv y^q \bmod p$). This attack will also work in the more general case that the authority knows a small multiple cq of q with $0 < c < w$; see [7, 8] for details.

Bleichenbacher's attack is particularly pernicious as it means that if an implementer chooses $g = 2$ for performance reasons, as we understand has happened, then signatures can be forged independently of the choice of p.

To thwart attacks of this kind, we can check that $(p-1)/\gcd(r, p-1)$ is not too smooth, or, once again, work in a subgroup of prime order. Another approach is to replace M by $h(r, M)$ as in the Schnorr signature [27, 28] (see [26]).

However, even using the DSS does not completely give complete protection, as we shall see below.

2.3 The design of the digital signature standard

During World War 2, a slogan prominently displayed in a number of the cryptanalysts' huts at Bletchley Park was 'If it's not checked, it's wrong' [15]. This prudent approach to cryptology is just as valid in the world of discrete logarithm based systems as it was in the days of rotor machines.

Consider what happens if the verifier's software does not systematically check whether $0 \leq r < p$ (this is not explicitly demanded in ElGamal's original paper [13]). A valid signature can easily be forged for any message as follows. Choose values for both s and r_{p-1} and compute

$$r_p = g^{\frac{M}{s}} y^{-\frac{r_{p-1}}{s}} \pmod{p}$$

Then it is straightforward to find a value r which fits the values of both r_{p-1} mod $p-1$ and r_p mod p by using the Chinese Remainder Theorem. Of course, the typical length of the r we obtain will be twice that of p. This attack has independently been discovered by Bleichenbacher [7, 8].

We conclude that DSS is simply ElGamal with many of the bugs fixed, and with a rather simple optimization (the reduction of r modulo q) that reduces the length of the signature and the amount of arithmetic needed to verify it.

However, DSS is not entirely bug-free. Firstly, the reduction mod q introduces a new bug: the authority can choose q to be the difference between $h(M)$ and $h(M')$ for two known messages M and M'. These two messages will then have the same signature [31]. Note that this kind of bug is avoided if M is hashed together with r, as in the Schnorr signature [27, 28].

Secondly, the authority might issue a base g with small order instead of order q. Honest signers would be unable to produce a valid signature, but it will be easy for anyone to forge one (by picking random signatures until one is valid with this g). So if the application may use a large number of different moduli and bases some of which come from non-trusted parties (as is the case with the UK government's new Euroclipper proposal [9]), then the cautious implementer will check that order of any base he uses for the first time. Other attacks based on the fact that g is not certified are also possible (see [31]).

The above two weaknesses probably do not matter in most digital signature applications, but it seems prudent practice to require a proof of origin of the p and q (for example, that they are generated using known seed values via a one-way hash function) and to check the order of g.

However, careless designs that permit degenerate values such as in these two examples can be deadly for Diffie Hellman key agreement protocols.

3 Diffie Hellman-like protocols

In the original Diffie Hellman Protocol, everything works modulo a fixed prime p. Two participants Alice and Bob first agree on this prime and a generator g of Z_p^*; then Alice picks a random a and sends $g^a \bmod p$ to Bob, who picks a random b and sends $g^b \bmod p$ to Alice. They then compute a shared private key as $g^{ab} \bmod p$.

3.1 The middleperson attack

In the traditional middleperson attack, Charlie sits between Alice and Bob. He sets up one key with Alice (pretending to be Bob), and another key with Bob (pretending to be Alice). He then acts as a message relay between them, and can intercept all their traffic. Attacks of this kind are a serious pitfall for designers of public key protocols, and can manifest themselves in various ways.

The traditional countermeasure is a key confirmation protocol in which Alice and Bob authenticate either the key they think they have just agreed, or the components they think they exchanged in order to agree it. We shall now show that these two countermeasures are not in fact equivalent; there are attacks in which Charlie forces Alice and Bob to share a key that he can calculate.

If a careless implementation does not check the received values, then Charlie can simply intercept both g^a and g^b and replace them with 0 or 1. Alice and Bob end up sharing a 'secret' that Charlie knows too.

Assuming as before that $p-1=qw$ with w smooth, a more sophisticated attack is for Charlie to replace the numbers g^a with g^{aq} and g^b with g^{bq}. In this way, the exchange is forced into the smooth subgroup of Z_p^*; he can then compute the discrete logarithm of $g^{aq} \bmod p$ to the base g^q and apply it to $g^{bq} \bmod p$, getting the shared key g^{abq}. This attack was discovered by van Oorschot and Wiener [32]; for more discussion on key agreement protocols, see Just and Vaudenay [18].

3.2 Attacks on elliptic curve systems

Other variants on Diffie Hellman use elliptic curves. A typical system uses curves of order $4p$, where p is prime [30]; in this case, an attacker can use the above techniques to force the key exchange into a subgroup of order 4, in which discrete logarithms may be extracted by inspection.

But regardless of whether we are operating in a rational or elliptic curve group, so long as its order is composite we have to be careful how we fortify the Diffie Hellman protocol against middleperson attacks (in which Charlie simultaneously masquerades as Bob to Alice and as Alice to Bob). We shall now discuss a few of the protocols that are vulnerable in this way.

3.3 Other Diffie-Hellman-like protocols

There are many variants of the Diffie-Hellman Protocol (see [12, 17, 21] for instance), which fail to specify that prime order bases should be used. We leave finding attacks to the reader as a exercise.

We also understand that in some secure telephone products, the two participants authenticate the key by reading out a hash of the agreed key. This is not sufficient against attacks of the kind described above, as the participants would end up authenticating $g^{abq} \bmod p$. Even if the base g were the generator of a prime order subgroup, then the telephone equipment should check that the received protocol variables were elements of order q, in case degenerate values had been inserted instead.

When trying to prevent such attacks, designers would be well advised to follow the principle that robust protocols are explicit ones [5]. As well as choosing a prime order base and checking received variables, we should also authenticate not just g^{ab} but also g^a, g^b, and all relevant environmental information such as Alice's and Bob's names and the date.

Mea culpa: one of us disregarded this principle in [4]. There, an authentication mechanism (of which the details are irrelevant here) was devised that enabled Alice and Bob to check whether they shared the same session key. This gives no protection against the above attacks.

4 Chor-Rivest-like Cryptosystems

At the beginning of public key cryptography, knapsack ciphers seemed very attractive, but almost all of them have been broken by latice reduction techniques (the Chor-Rivest system [10] remained unbroken until recently [29]).

We have noticed two recent attempts to make cryptosystems based on multiplicative knapsack problems.

4.1 The Lenstra Cryptosystem

The Lenstra Cryptosystem [19] is inspired by Chor-Rivest. It also works in a finite field $\mathrm{GF}(q^h)$, where q is a prime power. Chor-Rivest has the surprising property of being efficient when computing discrete logarithms in $\mathrm{GF}(q^h)$ is easy [10]. Lenstra's variant is a little less efficient but does not require this property. It is moreover provably at least as secure as Chor-Rivest.

Basically, given public parameters q, h and s (as well as the representation of the fields), the Lenstra Cryptosystem uses public keys $v_1, \ldots, v_s$ in $\mathrm{GF}(q^h)$. A message is first encoded (by using a public encoding rule) into a sequence $m_1, \ldots, m_s$ of positive integers such that $m_1 + \ldots + m_s = h$. Encrypting the message consists in computing ${v_1}^{m_1} \ldots {v_s}^{m_s}$ in $\mathrm{GF}(q^h)$. A complicated forgery algorithm (which involves secret keys for the v_i) lets us decrypt.

Here we notice that the order $q^h - 1$ of the multiplicative group may well be smooth (for instance, for each factor n of h, $q^n - 1$ is a factor of $q^h - 1$). Thus it may be possible to compute discrete logarithms in $\mathrm{GF}(q^h)^*$. But computing $w_i = \log v_i$ reduces the cipher to a knapsack equation $m_1w_1 + \ldots + m_sw_s$ (actually, a Chor-Rivest problem). Therefore (as mentioned by Lenstra) the ability to compute discrete logarithms reduces the security to exactly that of Chor-Rivest.

4.2 The Naccache-Stern Cryptosystem

A more complex example is given by a new public-key cryptosystem recently proposed by Naccache and Stern and based on multiplicative knapsacks [23, 25]. In this protocol, the public parameters are a set of prime numbers $p_1, \ldots, p_n$ and a prime p such that $p > p_1 \times \ldots \times p_n$. The secret key is an exponent s, and the public key is a set of integers $v_1, \ldots, v_n$ such that ${v_i}^s \equiv p_i \pmod{p}$. To encrypt an n-bit message $m_1 \ldots m_n$, we compute $c = {v_1}^{m_1} \times \ldots \times {v_n}^{m_n} \bmod p$. To decrypt the message, we raise c to the power s and solve $c^s = {p_1}^{m_1} \times \ldots \times {p_n}^{m_n} \bmod p$.

The authors already mentioned an information leakage in the cryptosystem. Namely, with the use of the Legendre symbol, the encryption leaks the bit

$$\left(\frac{c}{p}\right) = \left(\frac{v_1}{p}\right)^{m_1} \times \ldots \times \left(\frac{v_n}{p}\right)^{m_n} \tag{3}$$

This corresponds to the factor two in $p-1$. More generally, for any smooth factor w of $p-1$, one can compute the mod w part of the discrete logarithms of all the v_i and c: if g is a primitive w-th root of 1 modulo p, let ${v_i}^{\frac{p-1}{w}} \equiv g^{a_i} \pmod{p}$ and $c^{\frac{p-1}{w}} \equiv g^b \pmod{p}$. By computing all a_is and b, we obtain the equation

$$b \equiv m_1a_1 + \ldots + m_na_n \pmod{w} \tag{4}$$

This is a standard knapsack problem which can be solved by usual latice reduction tricks [16] provided that w is large enough (typically, larger than 2^n).

In [25], the authors propose using quadratic residues p_is to fix the problem of information leakage and to use only a strong prime p ($p = 2q + 1$) to avoid this kind of attack.

5 Conclusions

When designing systems based on the discrete logarithm problem, we must be careful to avoid degenerate cases. If we use a group with smooth subgroups, then there may be many such cases. In particular, naïve Diffie Hellman — whether modulo a rational prime or over an elliptic curve — can be attacked by forcing it into a subgroup in which the discrete log problem is easy. Some of the conventional ways of preventing middleperson attacks prevent such forcing attacks, but many do not.

Even without smooth subgroups, there are many things that can go wrong if we do not carefully specify the limit values and check them punctiliously at runtime. The system parameters, the secret keys and the message keys can often be chosen to be weak, and these weaknesses can interact in ways that are not at all obvious.

In any case, with all systems based on discrete logarithm problems, it seems prudent practice to use a group of prime order, unless there are good reasons not to; such reasons can include using keys for multiple different purposes. However, such advanced applications are dangerous, and designers should take extra care.

Our results have also explained the design of the US digital signature standard. It is really just ElGamal with most of the bugs fixed.

Finally, what 'good reasons' might there be to use a group of composite order? Well, different sets of people may know a secret key in different subgroups. We have already shown in [6] that this can be used to create broadband narrowcast subliminal channels in signature schemes — by means of which messages may be sent undetectably to precisely those parties who know a signing key modulo a specific factor of $p - 1$. Composite groups might also be used in applications such as key escrow protocols, where we might wish the same key to be used for escrowed encryption and unescrowed signature. However, this will be the subject of a different paper.

References

1. RJ Anderson, "Practical RSA Trapdoor", in *Electronics Letters* v 29 no 11 (27/5/93) p 995
2. RJ Anderson, "Why Cryptosystems Fail", in *Communications of the ACM* v 37 no 11 (Nov 94) pp 32–40
3. RJ Anderson, SJ Bezuidenhoudt, "On the Reliability of Electronic Payment Systems", in *IEEE Transactions on Software Engineering* v 22 no 5 (May 1996) pp 294–301
4. RJ Anderson, TMA Lomas, "On fortifying key negotiation schemes with poorly chosen passwords", in *Electronics letters* v **30** no 12 (23rd July 1994) pp 1040–1041
5. RJ Anderson, RM Needham, "Robustness principles for public key protocols" in *Advances in Cryptology — CRYPTO '95*, Springer LNCS v 963 pp 236–247
6. R Anderson, S Vaudenay, B Preneel, K Nyberg, "The Newton Channel", in *Preproceedings of the First International Workshop on Information Hiding* (30/5-1/6/96, Cambridge, UK) pp 143–148; proceedings to be published in Springer LNCS series
7. D. Bleichenbacher, "Generating ElGamal Signatures Without Knowing the Secret Key", in *Advances in Cryptology — Eurocrypt 96*, Springer LNCS v 1070 pp 10–18
8. D. Bleichenbacher, *'Efficiency and Security of Cryptosystems based on Number Theory'* Dissertation ETH No. 11404, Swiss Federal Institute of Technology, Zürich (1996)
9. "Securing Electronic Mail within HMG — part 1: Infrastructure and Protocol" 21 March 1996, CESG document T/3113TL/2776/11
10. B Chor, RL Rivest, "A knapsack-type public key cryptosystem based on arithmetic in finite fields", in *IEEE Transactions on Information Theory, v 34 (1988) pp 901–909*

11. Y Desmedt, P Landrock, A Lenstra, K McCurley, A Odlyzko, R Rueppel, M Smid, “The Eurocrypt 92 Controversial Issue — Trapdoor Primes and Moduli”, in *Advances in Cryptology — Eurocrypt 92*, Springer LNCS v 658 pp 194–199
12. W Diffie, PC van Oorschot, MJ Wiener, “Authentication and authenticated key exchanges”, in *Designs, Codes and Cryptography* v 2 (1992) pp 107–125
13. T ElGamal, “A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms”, *IEEE Transactions on Information Theory* v 31 no 4 (1985) pp 469–472
14. A Fiat, A Shamir, “How to prove yourself: practical solutions to identification and signature problems”, in *Advances in Cryptology — CRYPTO 86*, Springer LNCS v 263 pp 186–194
15. FH Hinsley, A Stripp, *‘Codebreakers’*, OUP 1993
16. A Joux, J Stern, “Lattice Reduction: a Toolbox for the Cryptanalyst”, to appear in *Journal of Cryptology*
17. N Jefferies, C Mitchell, M Walker, “A Proposed Architecture for Trusted Third Party Services”, in *Cryptography: Policy and Algorithms*, Springer LNCS v 1029 pp 98–104
18. M Just, S Vaudenay. “Authenticated multi-party key agreement”, *in these proceedings*
19. HW Lenstra, Jr., “On the Chor-Rivest Knapsack Cryptosystem”, in *Journal of Cryptology* v 3 (1991) pp 149–155
20. L Letham, D Hoff and A Folmsbee, “A 128K EPROM Using Encryption of Pseudorandom Numbers to Enable Read Access”, in *IEEE Journal of Solid State Circuits* v SC-21 (Oct 1986) pp 881 - 888
21. T Matsumoto, Y Takashima, H Imai. “On Seeking Smart Public-Key-Distribution Systems”. in *Transactions of the IECE of Japan* (1986) pp 99–106
22. D Naccache, “Unless modified Fiat-Shamir is insecure”, in *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography: SPRC 93*, Rome, Italy 15–16 Feb 1993 pp 172–180, *published by Fondazione Ugo Bordoni*
23. D Naccache, *‘Signature Numérique et Preuves à Divulgation Nulle, Cryptanalyse, Défense et Outils Algorithmiques’*, Thèse de Doctorat de l’Ecole Nationale Supérieure des Télécommunications ENST 95 E 019 (1995)
24. National Institute of Standards and Technology, *‘Digital Signature Standard’*, FIPS Publication 186 (19 May 1994)
25. D Naccache, J Stern, “A new public-key encryption scheme”, *presented at Luminy, September 1995*
26. D Pointcheval, J Stern. “Security proofs for signature schemes”, in *Advances in Cryptology — Eurocrypt 96*, Springer LNCS v 1070 pp 387–398
27. CP Schnorr, “Efficient identification and signature for smart cards”, in *Advances in Cryptology — CRYPTO 89*, Springer LNCS v 435 pp 239–252
28. CP Schnorr, “Efficient signature generation by smart cards”, in *Journal of Cryptology* v 4 (1991) pp 161–174
29. CP Schnorr, HH Hörner, “Attacking the Chor-Rivest Cryptosystem by improved lattice reduction”, in *Advances in Cryptology — Eurocrypt 95*, Springer LNCS v 921 pp 1–12
30. R Schroeppel, H Orman, S O’Malley, O Spatschek, “Fast Key Exchange with Elliptic Curve Systems”, in *Advances in Cryptology — Crypto 95*, Springer LNCS v 963 pp 43–56

31. S Vaudenay, “Hidden collisions on DSS”, in *Advances in Cryptology — CRYPTO ’96*, Springer LNCS v 1109 pp 83–88
32. P. van Oorschot, M. J. Wiener, “On Diffie-Hellman key agreement with short exponents”, in *Advances in Cryptology — Eurocrypt 96*, Springer LNCS v 1070 pp 332–343