Breaking a Cryptographic Protocol with Pseudoprimes

Daniel Bleichenbacher

Bell Labs, Lucent Technologies

Abstract. The Miller-Rabin pseudo primality test is widely used in cryptographic libraries, because of its apparent simplicity. But the test is not always correctly implemented. For example the pseudo primality test in GNU Crypto 1.1.0 uses a fixed set of bases. This paper shows how this flaw can be exploited to break the SRP implementation in GNU Crypto. The attack is demonstrated by explicitly constructing pseudoprimes that satisfy the parameter checks in SRP and that allow a dictionary attack. This dictionary attack would not be possible if the pseudo primality test were correctly implemented.

Often important details are overlooked in implementations of cryptographic protocols until specific attacks have been demonstrated. The goal of the paper is to demonstrate the need to implement pseudo primality tests carefully. This is done by describing a concrete attack against GNU Crypto 1.1.0. The pseudo primality test of this library is incorrect. It performs a trial division and a Miller-Rabin test with a fixed set of bases. Because the bases are known in advance an attacker can find composite numbers that pass the primality test with probability 1. A protocol implemented in GNU Crypto that requires a reliable primality test is SRP. The security of SRP depends on a group for which computing DL is hard. In SRP the server chooses the group parameters and sends them to the client. It is then important that the client verifies that computing DLs in the chosen group is indeed hard. Otherwise, the client could expose his password to a dictionary attack. This paper shows that the flaw in the GNU Crypto primality test indeed weakens the SRP implementation by explicitly constructing weak parameters for SRP. The weakness would not exist if a reliable primality test were implemented.

1 The Miller-Rabin pseudo primality test.

A well-known Theorem by Fermat states that if n is a prime and b is coprime to n then

$$b^{n-1} \equiv 1 \pmod{n} \tag{1}$$

Hence if Equation (1) is not satisfied for a pair (b, n) that is coprime then n is composite. Unfortunately, there also exist pairs (b, n) that satisfy Equation (1), but where n is composite. Composite numbers n that satisfy Equation (1) for all

b coprime to n are called Carmichael numbers. Korselt proposed the following criterion for such numbers [7].

Korselt's criterion. A composite number n is a Carmichael number if and only if n is squarefree and all prime divisors p of n satisfy

$$p - 1 | n - 1.$$

Because of the existence of Carmichael numbers [4] Equation (1) alone cannot be used to distinguish composites from primes. Miller and Rabin proposed a stronger test based on the following observation. Let n be an odd integer and write $n = u2^v + 1$ with u odd. Then for every odd prime n and every base $1 \leq b < n$ one of the following two conditions is satisfied:

$$b^u \equiv 1 \pmod{n} \tag{2}$$

or there exists $0 \leq i < v$ such that

$$b^{u2^i} \equiv -1 \pmod{n} \tag{3}$$

A composite n is called a *strong pseudoprime* for the base b if one of the conditions is satisfied. Rabin showed that if n is composite n is a strong pseudoprime for less than $n/4$ bases $b \in [2, n-1]$ [10]. Thus any composite number n can be recognized as composite with probability at least $1 - (1/4)^k$ by selecting k random bases and testing whether n fails the test for at least one base.

Damgard, Landrock, and Pomerance prove a bound much lower than $(1/4)^k$ on the average probability that a composite number passes a Miller-Rabin test with k bases [5]. This result, however, cannot be applied in cryptographic protocols for a parameter verification. If a party has to verify that a received integer is prime, then the party should assume the worst case, i.e., that the integer might have been chosen to maximize the probability of passing a Miller-Rabin test.

2 SRP

GNU Crypto implements SRP-6 [11]. The goal of the SRP protocol is to avoid offline dictionary attacks and thus increase the security of password based authentications in the case that the clients password has not much entropy. In particular, a server that does not know the clients password or a value v, which is derived from it should only be able to confirm or reject one password guess per login. This section reviews one version of the SRP protocol and describes why it is important that the client performs a proper parameter verification in step 2.

Client		Server
1.	$\xrightarrow{I}$	(lookup s, v, g, N)
2. (verify g, N) $x = \mathrm{H}(s, I, P)$	$\xleftarrow{s,g,N}$	
3. (choose random a)		(choose random b)
4. $A = g^a \bmod N$	$\xrightarrow{A}$	$B = 3v + g^b \bmod N$
5. $u = \mathrm{H}(A, B)$	$\xleftarrow{B}$	$u = \mathrm{H}(A, B)$
6. $S = (B - 3g^x)^{a+ux} \bmod N$		$S = (Av^u)^b \bmod N$
7. $M_1 = \mathrm{H}(A, B, S)$	$\xrightarrow{M_1}$	(verify M_1)
8. (verify M_2)	$\xleftarrow{M_2}$	$M_2 = \mathrm{H}(A, M_1, S)$
9. $K = \mathrm{H}(S)$		$K = \mathrm{H}(S)$

In step 1 the client sends its identity I to the server and the server looks up the corresponding values s, v, g, N, where s is a salt value, g is a generator of $\mathbb{Z}/(N)^*$ and v is encryption of the clients password defined by $v = g^{\mathrm{H}(s,I,P)} \bmod N$.

In step 2 s and optional g and N are sent to the client. The client must verify that N is a strong prime $> 2^{512}$, i.e. N and $(N-1)/2$ are both prime and that g has order $N-1$ in $\mathbb{Z}/(N)^*$.

In step 3 both client and server choose some random values a and b respectively and derive two values A and B, which are then exchanged in step 4 and 5. Both server and client can now compute a mutual secret S. This value S is subsequently used for a mutual authentication in step 7 and 8.

An outsider, or even a malicious server not knowing v should not be able to verify the correctness of a guessed password P from the values observed during a protocol run.

Attacking SRP with bogus parameters. MacKenzie noticed that Tom Wu's SRP implementations before version 1.6.0 are susceptible to an offline dictionary attack [8]. In particular, MacKenzie noticed that while the SRP documentation requires that N and $(N-1)/2$ are primes the implementation does not perform any primality checks when a client receives new parameters from a server. But these checks are crucial for the protocol.

If an attacker posing as a server is able to submit parameters g, N, such that computing the discrete logarithm of $g^x \bmod N$ is computable then the following attack is possible.

Client		Attacker
1.	$\xrightarrow{I}$	(select s, g, N)
2. (verify g, N) $x = \mathrm{H}(s, I, P)$	$\xleftarrow{s,g,N}$	
3. (choose random a)		
4. $A = g^a \bmod N$	$\xrightarrow{A}$	(choose any B)
5. $u = \mathrm{H}(A, B)$	$\xleftarrow{B}$	$u = \mathrm{H}(A, B)$
6. $S = (B - 3g^x)^{a+ux} \bmod N$		
7. $M_1 = \mathrm{H}(A, B, S)$	$\xrightarrow{M_1}$	(abort)

Hence after aborting the protocol in step 7 the attacker has now enough information for an offline dictionary attack. SRP was designed to prevent such attacks. From the assumption that DLs mod N are computable follows that the server can compute a such that $g^a = A \bmod N$. Now, the attacker can perform an offline dictionary attack by first guessing P', computing $x' = \mathrm{H}(s, I, P)$ and $S' = (B - 3g^x)^{a+ux} \bmod N$. Finally if $\mathrm{H}(A, B, S')$ equals M_1 then P' is likely the client's password.

3 GNU Crypto

An analysis of the primality test in GNU Crypto 1.1.0 shows a serious flaw. The primality test, first performs a trial division test and then calls the routine `gnu.util.prime.passEulerCriterion`. This routine is a Miller-Rabin with the primes up to 41 as bases. Since the bases are fixed it is possible to find counter examples that pass the test with probability 1.

4 Constructing pseudoprimes

Requirements. Composite numbers that pass the GNU Crypto 1.1.0 primality test are well known. For example Arnault has previously constructed a composite 337 digit number that is a strong pseudoprime for the 200 smallest prime bases [3]. The construction that Arnault used generates integers that are the product of a small (i.e. 2 or 3) number of large primes. While these number would incorrectly pass the parameter checks they cannot be used to break SRP.

The goal of this paper is to find parameters that pass the checks for the SRP in GNU Cryptos implementation and allow a server to find a users password. This requires to construct a triple (g, N, q) such that computing discrete logarithms of $g^x \pmod{N}$ is easy, $N = 2q + 1 > 2^{512}$, both N and q pass the primality test, $N > 2^{512}$, and $g^q \equiv -1 \pmod{N}$.

The construction given in this paper constructs q such that it is the product of small primes. Then computing DLs modulo $N = 2q + 1$ is easy, because the algorithm by Pohlig and Hellman [9] can be applied.

Description of the method. The method used here is based on an idea by Erdös [6] to estimate the distribution of Carmichael numbers. Erdös suggested to construct Carmichael numbers as follows. First choose an even integer M that has many divisors. Let R be the set of primes r such that $r-1$ is a divisor of M. If a subset $T \subset R$ can be found such that

$$C = \prod_{r \in T} r \equiv 1 \pmod{M}, \tag{4}$$

then C is a Carmichael number, because C satisfies Korselt's criterion. One can hope to find such sets T if R contains more than about $\log_2 M$ primes.

Erdös estimates were only heuristic. But Alford, Granville and Pomerance extended his idea and were able to prove that there exist infinitively many Carmichael numbers [2]. The main difficulty of this proof was to show that for suitably chosen integers M the corresponding set of primes R is large enough to guarantee that Equation 4 can be solved for a subset $T \subset R$.

Additionally, a Carmichael number C is a strong pseudoprime for a base b if the order of b modulo r is divisible by the same power of 2 for all primes factors r of C. If all prime factors r are congruent 3 modulo 4 then this condition is satisfied when b is a quadratic residue modulo either all prime factors r or none at all, because in that case the order of b modulo r is either even or odd for all r. In particular, it is possible to construct a Carmichael number that is strong pseudoprime for a set of bases B as follows: Choose a suitable integer M. Then find a set R of primes, such that for all bases $b_i \in B$ there exists $c_i \in \{-1, 1\}$ with $\left(\frac{b_i}{r}\right) = c_i$ for all $r \in R$. Finally, find a subset $T \subset R$ can be found that satisfies Equation 4.

The results by Alford, Granville and Pomerance are strong enough to show that even under these restrictions large enough sets R can be found. In particular, they showed the existence of infinitively many counter examples to a Miller-Rabin test with a fixed set of bases [1].

To pass the parameter checks in GNU Crypto the pseudoprime C needs the additional property that $2C+1$ is prime or pseudoprime. Because of this additional property it appears difficult to prove the existence of counter examples for arbitrary sets of bases.

However, the goal of this paper is to construct a pseudoprime for a given set of bases only, i.e. the set $B = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41\}$ that is used in GNU Crypto. To do so let

$$M = 2 \cdot 5^3 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 61.$$

Next a set R of all integers r satisfying

$$\begin{aligned} 256 < \ & r < 2^{60}, \\ r - 1 \ & | \ M, \\ & r \quad \text{is prime}, \\ \left(\frac{b_i}{r}\right) & = c_i \text{ for all } 1 \le i \le 13, \end{aligned}$$

where the pairs (b_i, c_i) are defined as follows:

i	1	2	3	4	5	6	7	8	9	10	11	12	13
b_i	2	3	5	7	11	13	17	19	23	29	31	37	41
c_i	-1	1	1	-1	-1	1	1	-1	-1	1	-1	1	1

The values c_i should are chosen in such a way that $\left(\frac{b_i}{r}\right) = c_i$ is possible for primes $r \equiv 1 \pmod{b_i}$. The set R can be found efficiently, by first constructing all divisors d of M and checking if $r = d + 1$ satisfies the remaining conditions.

The set of integers satisfying all these conditions contains 64 primes $R = \{r_1, \ldots r_{64}\}$. Next find subsets $T \subset R$ with at least 2 elements satisfying Equation 4, i.e., $\prod_{r \in T} r \equiv 1 \pmod{M}$. These subsets T can be found using a meet-in-the-middle approach. I.e., R is divided into two distinct subsets R_1 and R_2. The values $(\prod_{r \in T_1} r)^{-1} \bmod M$ are precomputed for all $T_1 \subseteq R_1$ and stored in a table. Then for all $T_2 \subseteq R_2$ the value $\prod_{r \in T_2} r \bmod M$ is computed. If this value is contained in the table then set $T = T_1 \cup T_2$ and $C = \prod_{r \in T} r$. If furthermore $N = 2C + 1$ is prime and $N > 2^{512}$ then N passes the parameter test for SRP in GNU Crypto. This is shown in the next paragraph.

Correctness of the construction. Since N is prime it remains to show that C passes the primality test. The assumption $256 < r$ implies that no prime factor of C is found during the trial division test. Thus it is sufficient to show that C is a strong pseudoprime for the bases b_i where $1 \le i \le 13$.

Since $r - 1$ divides M and $C \equiv 1 \pmod{M}$ it follows that $r - 1$ divides $C - 1$ for all prime factors r of C. Thus by Korselt's criterion C is a Carmichael number [7]. Because of $\left(\frac{2}{r}\right) = -1$ we have $r \equiv 3 \pmod{4}$ for all primes factor r of C. Moreover, from $\left(\frac{b_i}{r}\right) = c_i$ for all $1 \le i \le 13$ follows that b_i is either a quadratic residue for all prime factors r or a quadratic nonresidue for all prime factors r. Hence it follows that C is a strong pseudoprime for the base b_i.

Results. An implementation of the algorithm needed less than 10 days on a 250 MHz CPU to find about 30 examples that pass the parameter checks in GNU crypto. One example is the 1095 bit number

$$\begin{gathered} C = 398462957079251 \cdot 28278016308851 \cdot 268974870654491 \cdot 1239515532971 \cdot \\ 12941222544251 \cdot 2825874899 \cdot 182200861571 \cdot 480965007251 \cdot 8028415890251 \cdot \\ 761874633627251 \cdot 10326412038251 \cdot 105324823451 \cdot 7128348371 \cdot 29542620251 \cdot \\ 251906132167691 \cdot 64654312451 \cdot 226698699371 \cdot 130685132579 \cdot 9167201891 \cdot \\ 432876391197251 \cdot 3077983389251 \cdot 17767646051 \cdot 9371850251 \cdot 954045342251 \cdot \\ 112810627931 \cdot 6297653304192251 \cdot 20842025454251 \end{gathered}$$

5 GNU Crypto 2.0.1

The authors of GNU Crypto were informed in January 2004 about the flaws in the primality test. Most of the problems have been fixed in version 2.0.1. However, an analysis of the source code reveals that GNU Crypto implementation of SRP still calls the function `gnu.util.prime.passEulerCriterion` and that this function has not been changed. Therefore the attack presented in this paper still exists more than 8 month after the authors have been notified. The next implementation error can be found just 2 lines later where SRP accepts $g \equiv -1 \pmod{N}$ as a generator of $\mathbb{Z}/(N)^*$ allowing a simple impersonation attack. Consequently, I do not recommend the use of GNU Crypto.

6 Proposed parameter verification for SRP

To verify that $N > 2^{512}$ is a safe prime (that is both N and $q = (N-1)/2$ are prime) and g is a generator of $\mathbb{Z}/(N)^*$ with an error probaility $< 2^{-2k}$ one can perform the following tests:

- Check $N > 2^{512}$.
- Test the primality of q with k rounds of Miller-Rabin with random bases.
- Test that $1 < g < N-1$ and $g^q \equiv -1 \pmod{N}$.

The k rounds of Miller-Rabin guarantee that a composite q is detected with a probability $> 1-2^{-2k}$. Assuming that q is indeed prime $g^q \equiv -1 \pmod{N}$ now implies that the order of g modulo N is even and divides $2q$. Hence the order is either 2 or $2q$. But $g^2 \equiv 1 \pmod{N}$ would imply $g \equiv g^q \equiv -1 \pmod{N}$ which is impossible because of $1 < g < N-1$. Thus the order of g must be $2q$ and $N = 2q+1$ must be prime. Hence no primality test for N is needed here.

References

1. W. R. Alford, A. Granville, and C. Pomerance. On the difficulty of finding reliable witnesses. In *Algorithmic number theory*, volume 877 of *Lecture Notes in Computer Science*, pages 1–16, Berlin, 1994. Springer Verlag.
2. W. R. Alford, A. Granville, and C. Pomerance. There are infinitely many Carmichael numbers. *Annals of Mathematics*, 140(3):703–722, 1994.
3. F. Arnault. Rabin-Miller primality test: Composite numbers which pass it. *Mathematics of Computation*, 64(209):355–361, Jan. 1995.
4. R. D. Carmichael. On composite numbers P which satisfy the Fermat congruence $a^{P-1} \equiv 1 \bmod P$. *American Math. Monthly,*, 19:22–27, 1912.
5. I. Damgård, P. Landrock, and C. Pomerance. Average case error estimates for the strong probable prime test. *Mathematics of Computation*, 61(203):177–194, 1993.
6. P. Erdös. On pseudoprimes and Carmichael numbers. *Publ. Math. Debrecen*, 4:201–206, 1956.
7. A. Korselt. Problème chinois. *L'intermédiaire des mathématiciens*, 6:142–143, 1899.
8. P. MacKenzie. Personal communications.
9. S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT-24:106–110, Jan. 1978.
10. M. Rabin. Probabilistic algorithms for testing primality. *J. Number Theory*, 12:128–138, 1980.
11. T. Wu. SRP-6: Improvements and refinements to the secure remote password protocol. URL=`http://srp.stanford.edu/doc.html`, Oct. 2002.