User Impersonation in Key Certification Schemes

Arjen K. Lenstra and Yacov Yacobi
Bellcore, 445 South Street, Morristown, NJ 07960, U.S.A.
lenstra@bellcore.com yacov@bellcore.com

Communicated by Gilles Brassard

Received 22 October 1991 and revised 4 September 1992

Abstract. In this note we exhibit some weaknesses in two key certification schemes. We show how a legitimate user can impersonate any other user in an ElGamal-based certification scheme, even if hashing is applied first. Furthermore, we show how anybody can impersonate users of the modular square root key certification scheme, if no hashing occurs before the certification. This shows that it is essential for this certification scheme to hash a message before signing it.

Key words. Public key, Key certification, Certification authority.

1. Introduction

A certificate of a public key is a proof that a linkage exists between a user's name and his public key. This proof is usually achieved using a signature of some trusted authority on a function of a user's name and his public key.[1] In this paper we present several weaknesses that affect the security of two key certification schemes that have appeared in the literature.

In [3] a key certification scheme is proposed that is based on ElGamal's signature scheme [4]. We show that legitimate users of this scheme who satisfy certain weak requirements, can be impersonated by other legitimate users. Although the identity of the impersonator can probably be detected if the impersonation is discovered, this is clearly a severe weakness in the key certification scheme from [3].

Another disadvantage of this scheme is the fact that the verification operation requires several modular exponentiations, and is thus fairly slow. In practical circumstances we would like to have an easily verifiable certificate, since verification is frequently done and in many cases by weak processors, such as can be found in smart-cards or portable phones. The creation of a certificate, on the other hand, may have higher complexity, since it is less frequently done and the trusted authority, who creates the certificate, may have extensive computational power.

[1] If identity-based systems are used, then there is no need for additional certificates.

This asymmetry leads to public-key signature schemes such as the modular square root scheme, proposed by Rabin [9], where the verification operation is a single modular squaring. For efficiency, it is common to hash a long message before signing it. For the modular square root key certification scheme, which is Rabin's modular square root signature scheme applied to the concatenation of a user's name and his public key, hashing is not important to achieve efficiency. It follows from [2] and [11], however, that for this scheme hashing is important for security reasons: even a complete outsider can impersonate any user of the modular square root key certification scheme, if no hashing takes place before certifying. To foil the attacks from [2] and [11] almost any hashing function suffices. For instance, interleaving the bytes of the user's name and his public key will do. Hashing does not help to foil our attack on the ElGamal-based scheme.

We present an alternative attack on the modular square root key certification scheme, which is similar to the attack from [2] and considerably simpler than the one from [11]. Although, like the attacks from [2] and [11], our attack might be detected by a suspicious verifier, it allows a variation which was not considered by [2] and which cannot, as far as we know, be achieved using [11]. Like the other attacks, our attack is foiled by hashing.

The attacks that we propose do not require any heavy machinery. On the contrary, they are totally straightforward and most of them take only a few seconds if the basic extended precision integer arithmetic routines are available. This raises another neglected issue. In the theories we develop the legitimate protocols ought to run in polynomial time, while adversarial computations should take super-polynomial time if they want to stand a chance of breaking the protocols. However, a theoretically clean successful attack against a given protocol should run in polynomial time, and it should not be detectable by any polynomial-time computation that the victim may do on top of the bare-bones protocol. Our attacks achieve these objectives except for the last: as we will see below a suspicious victim might do some extra work and detect our fraud in polynomial time.

The remainder of this paper is organized as follows. In the next section we discuss the ElGamal-based certification scheme and show how its users can be impersonated. The modular square root certification scheme is similarly treated in the third section.

Throughout this paper, a mod n denotes the last nonnegative remainder of a modulo n, for $a \in \mathbf{Z}$ and $n \in \mathbf{Z}_{>0}$.

2. The ElGamal-Based Certification Scheme

2.1. *ElGamal's Scheme*

In [4] ElGamal proposed the following public-key signature scheme. The public key of the universally trusted authority consists of three parts: a large prime number p, a generator g of the multiplicative group $\mathbf{F}_p^*$ of units of the finite field $\mathbf{F}_p$ of p elements, and some element $Y \in \mathbf{F}_p^*$. The field $\mathbf{F}_p$ is identified with the set $\{0, 1, \ldots, p-1\}$ in the "natural" way. A signature for a message $m \in \mathbf{Z}$ is a pair of

integers C_m, V_m such that

$$g^m \equiv Y^{C_m} \cdot C_m^{V_m} \mod p, \tag{2.1}$$

using the above mapping between field elements and integers. In practice, it is probably better to hash the original message to an m with $0 \leq m < p-1$. To compute the signature as in (2.1), the authority employs the secret information it has about its public key, namely, the discrete logarithm X with respect to g of Y, i.e., $X \in \mathbf{Z}$ such that $Y \equiv g^X \bmod p$ and $0 \leq X < p-1$. This can be achieved as follows. First, the trusted authority selects a random integer R_m uniformly from the interval $[1, p-2]$, such that R_m and $p-1$ are relatively prime. Next, C_m is set equal to the "integer" $g^{R_m} \bmod p$ (again, using the above-mentioned identification), and $R'_m \in \mathbf{Z}$ is computed, using, for instance, the extended Euclidean algorithm, such that $R_m \cdot R'_m \equiv 1 \bmod (p-1)$. Finally, V_m is set equal to $((m - X \cdot C_m) \cdot R'_m) \bmod (p-1)$.

Given the triple (m, C_m, V_m) anybody can verify that (2.1) holds: since $m \equiv X \cdot C_m + R_m \cdot V_m \bmod (p-1)$, we have that $g^m = g^{X \cdot C_m} \cdot g^{R_m \cdot V_m} \equiv Y^{C_m} \cdot C_m^{V_m} \bmod p$. Notice that any message $m \in \mathbf{Z}$ can be signed using this scheme, unlike the scheme to be discussed in the next section. Clearly, anyone who can solve the discrete logarithm problem in $\mathbf{F}_p$ can forge signatures, which raises the issue of the integrity of the public key (see Section 4). It is as yet unknown whether forging signatures is as hard as the discrete logarithm problem, or the Diffie–Hellman problem. In any case, we are not aware of an efficient way to forge ElGamal signatures, i.e., to produce a valid signature C_m, V_m for an arbitrary message m, when the public key is chosen properly.

2.2. *Key Certification Using ElGamal's Scheme*

There is an application of this signature scheme, however, for which it is not at all difficult to forge certificates, if one certificate is known. This concerns a key certification scheme that was mentioned in [3], and which has never been implemented [8]. Suppose that the trusted authority has public key p, g, Y, as described above. Furthermore, suppose that the name of user A, plus other personal data relevant for a unique identification of A, is coded in the positive integer I_A. We assume that A is known by this public identity I_A.

To become a legitimate user of the system, user A selects a secret positive integer X_A, computes the corresponding public key $Y_A = g^{X_A} \bmod p$, and presents the pair I_A, Y_A to the trusted authority for certification. To certify the linkage between I_A and Y_A, the trusted authority computes $m_A = Y_A^{I_A} \bmod p$, applies the ElGamal signature scheme to m_A, and provides A with the resulting certificate $C_A = C_{m_A}$, $V_A = V_{m_A}$. Someone who gets (I_A, Y_A, C_A, V_A) has to carry out four exponentiations modulo p to check that $g^{(Y_A^{I_A})} \equiv Y^{C_A} \cdot C_A^{V_A} \bmod p$, thus proving the linkage between A's name I_A and A's public key Y_A.

2.3. *Weakness in the ElGamal Key Certification Scheme*

The certification scheme in Section 2.2 has at least one major flaw. Suppose that user A with identity I_A went through the above-described procedure, and got

everything a legitimate user of the system should have: the secret key X_A, the corresponding public key Y_A, and the certificate C_A, V_A. For any integer s that is coprime to $p-1$, user A can compute $I_{\tilde{A}} = I_A/s \bmod(p-1)$ and $Y_{\tilde{A}} = Y_A^s$, and claim that $(I_{\tilde{A}}, Y_{\tilde{A}}, C_A, V_A)$ certifies that the user whose identity is $I_{\tilde{A}}$ has public key $Y_{\tilde{A}}$; furthermore, A easily computes the secret information $X_{\tilde{A}} = X_A \cdot s \bmod(p-1)$ that corresponds to this forgery. Any user B with identity I_B for which $(I_A/I_B) \bmod(p-1)$ is well defined can thus be impersonated by A, simply by using $s = (I_A/I_B) \bmod(p-1)$.

Hashing $Y_A^{I_A}$ clearly does not affect this impersonation attack. A disadvantage of the attack is, however, that the impersonator A is detectable: a victim (i.e., a verifier) who discovers that he is being cheated, informs the trusted authority who then traces the forgery back to the probably very small set of users with certificate C_A, V_A.

3. The Modular Square Root Certification Scheme

3.1. *Rabin's Scheme*

In [9] Rabin proposed the following public-key signature scheme. The public key of the universally trusted authority consists of a composite number $N \in \mathbf{Z}_{>0}$. The ring of integers $\mathbf{Z}/N\mathbf{Z}$ is identified with the set $\{0, 1, \ldots, N-1\}$ in the "natural" way. A signature for a message $m \in \mathbf{Z}$ is an integer C_m such that

$$m \equiv C_m^2 \mod N. \tag{3.1}$$

Evidently, this equation only has a solution if m is a square modulo N; we may assume that m consists of a hash of the original message concatenated with a few bits to make it a square modulo N. To compute the signature, and to decide which bits to concatenate,the trusted authority uses the secret information it has about N, i.e., N's factorization (see [7]).

Rabin [9] showed that taking square roots modulo N is as hard as factoring N. With the proper choice of N and without a further breakthrough in factoring techniques, it therefore seems to be infeasible to forge signatures, and we have no intention of refuting this assumption.

3.2. *The Modular Square Root Certification Scheme*

In an early version of [1] the following key certification application of Rabin's scheme was proposed. Suppose that the trusted authority has public key N as in Section 3.1, and that user A has public identity I_A as in Section 2.2. Furthermore, let the bitstring N_A be A's public key. Often, N_A will be identified with the positive integer represented by its bits. As usual, it is assumed that A possesses some additional secret information about the public key N_A that enables A to carry out a certain protocol based on N_A that cannot be carried out by parties that do not have this additional information about N_A. For example, N_A can be the product of two large primes that are only known to A; this enables A to compute square roots modulo N_A, which is commonly believed to be computationally infeasible for parties who do not know N_A's factorization.

To get a certificate for the pair I_A, N_A, user A presents $h(I_A, N_A)$ to the trusted authority. Here h is some simple publicly known and length-preserving function of I_A and N_A: it might simply concatenate I_A and N_A in any order, interleave their bits, or hash them in any other length-preserving way; in [1] the bits of I_A and N_A were simply concatenated. The trusted authority then employs its secret information about N to select a short bitstring w such that the concatenation $(h(I_A, N_A), w)$ of $h(I_A, N_A)$ and w is a quadratic residue modulo N, and provides A with one of the square roots C_A of $(h(I_A, N_A), w)$ modulo N. Someone who gets the triple (I_A, N_A, C_A) needs to perform only h and one modular squaring to verify the linkage between A's name I_A and A's public key N_A. Without loss of generality we assume that w has length 0, so we omit w in the following.

3.3. *Weaknesses in the Modular Square Root Certification Scheme*

We suppose that the protocol based on N_A is such that, for some random bit string $\tilde{N}_A$ of approximately the same length as N_A, there is a fair probability that the protocol based on $\tilde{N}_A$ can be carried out efficiently (possibly after deriving the "secret" information about $\tilde{N}_A$, which might not be hard to find for random $\tilde{N}_A$). Notice that the example mentioned in Section 3.2 satisfies this requirement. Furthermore, we assume that $\log_2 I_A$ is smaller than the length of N_A, say by a factor of at least 2.

Under these assumptions we can, for any I_A, construct $\tilde{C}_A$ and $\tilde{N}_A$ such that $\tilde{C}_A$ is a certificate for $h(I_A, \tilde{N}_A)$, and such that we are able to carry out the protocol based on $\tilde{N}_A$. This enables us to impersonate user A. Our attack will be successful if our potential victim, the verifier, *only* inspects the triple $(I_A, \tilde{N}_A, \tilde{C}_A)$ by certifying that indeed $\tilde{C}_A^2 \equiv h(I_A, \tilde{N}_A) \bmod N$. Our foiling scheme breaks down, however, if the verifier becomes suspicious and has a much closer look at the forged public key $\tilde{N}_A$. We are not aware of any applications that do more than the basic verification.

Our attacks work by constructing *many* different triples $(I_A, \tilde{N}_A, \tilde{C}_A)$ with $\tilde{C}_A^2 \equiv h(I_A, \tilde{N}_A) \bmod N$, where the $\tilde{N}_A$ behave, more or less, like random bitstrings of the proper length. For the example in Section 3.2, where we need to know the factorization of $\tilde{N}_A$ in order to impersonate user A successfully, it suffices to try until one of the $\tilde{N}_A$'s can be easily factored (for instance, by being prime). A suspicious verifier might notice such an easy $\tilde{N}_A$, but for practical protocols on small machines it is too costly to check that $\tilde{N}_A$ is indeed composite and difficult to factor. Furthermore, we might spend quite some time on our impersonation attack for some particularly interesting user A, and try until we have found an $\tilde{N}_A$ that has, for instance, one large and one medium-sized factor of, say, 25–35 digits, that can only be found after a fairly serious elliptic curve attack. The density of candidate $\tilde{N}_A$'s is sufficiently high for this approach to work.

We present three types of attack, depending on three different functions h of I_A and N_A: functions that arbitrarily interleave the bits of I_A and N_A in Section 3.4, bit-wise concatenation of I_A and N_A, with I_A on the left, in Section 3.5, and finally concatenation with I_A on the right in Section 3.6. We will see that the function from Section 3.5 is most susceptible to impersonation attacks. This variant was also considered in [2] and [11]; neither of these papers considers the variants from Sections 3.4 or 3.6.

Suppose that N has n bits (i.e., $2^{n-1} \leq N < 2^n$) and that I_A, the identity of the user we want to impersonate, has m bits. Because the concatenation of I_A and the public key N_A must be a positive integer less than N, we find that the integer N_A has at most $n - m$ bits; without loss of generality we may assume that this integer has $n - m$ bits.

3.4. *Attack on Arbitrary h*

If h arbitrarily interleaves the bits of I_A and N_A, we simply try positive integers $\tilde{C}_A$ at random, until $\tilde{C}_A^2 \bmod N = h(I_A, \tilde{N}_A)$, for some $\tilde{N}_A$ that is *feasible*, i.e., a number for which A's protocol can be carried out efficiently. Although the numbers that we generate in this way do not entirely behave as random numbers, it is not unreasonable to expect that the number of trials required will be close to 2^m divided by the probability that the $n - m$ bit number is feasible. Depending on the protocol, and for small m, say up to 20, this attack might work. Evidently, it is not a polynomial-time attack.

However, we do not have to try until a particular name I_A shows up, we can simply try until *any* name, or any name in a certain class of targets, shows up. Depending on the density of (target) names among the m-bit numbers, this might drastically increase the success rate of this approach. For instance, attacks to get access to the computing facilities of a major company or of a major network might use this approach (and actually have used a similar approach [10]).

Notice that the above attack works irrespective of how h intermingles the bits of I_A and N_A. It follows that in this type of key certification scheme in the first place m should be sufficiently large. Secondly, the probability to hit any name should also be kept as low as possible, which means that the number of legitimate identities should be very small compared with 2^m. A one-way h would also be sufficient to foil this attack.

3.5. *Identity Concatenated to the Left*

Let $h(I_A, N_A) = 2^{n-m} I_A + N_A$, i.e., the identity appears in the m most significant bits of the message to be signed. Without loss of generality we assume that $h(I_A, N_A) < N$. Let x be a random $n - 2m - 4$ bit integer such that the interval $[xN + 2^{n-m} I_A + 2^{n-m-1}, xN + 2^{n-m} I_A + 2^{n-m} - 1]$ contains at least one perfect square; this condition can be trivially checked, and it can be easily seen that most $n - 2m - 4$ bit x's satisfy it. For any such x we can compute the plain "real" square root, rounded downward, $C_x = \lfloor \sqrt{xN + 2^{n-m} I_A + 2^{n-m} - 1} \rfloor$. It follows from the condition on x and I_A that $C_x^2 \bmod N$ equals $h(I_A, N_x)$ for some integer N_x that satisfies $2^{n-m-1} \leq N_x < 2^{n-m}$. Selection of x's is repeated until N_x is feasible, in which case we put $\tilde{C}_A = C_x$ and $\tilde{N}_A = N_x$. The expected number of random x's needed before this happens depends of course on the protocol that is used. For the example mentioned in Section 3.2 the search-space is more than rich enough to make this approach successful, under the usual assumptions concerning the random behavior of the numbers involved. Notice that $\tilde{C}_A$ can have length up to $n - m - 2$, which we call a *forgery of length* $n - m - 2$. Even a very naive verifier might notice that the certificate is quite short, also if we disguise $\tilde{C}_A$ by using $N - \tilde{C}_A$ instead.

This problem of short forgeries is overcome by the more complicated approach from [11] which can be used under the reasonable assumption that $m < n/3$ (see Section 3.3). Using a method based on lattice basis reduction, the following is shown in [11]: given almost any bitstring x of length $n - m$ with $2^m x + 2^{n-m} - 1 < N$ and $m < n/3$, one can efficiently compute C_x with $0 \le C_x < N$ such that the most significant $n - m$ bits of C_x are equal to x, and such that $C_x^2 \bmod N$ consists of the concatenation of I_A and an $n - m$ bit integer N_x. For details we refer to [11]. This construction, which is much more general than ours but requires considerably more computing time, can again be repeated until a feasible N_x is found. If this impersonation attack is used, only the much more suspicious verifier who also inspects the foiled public key will notice the forgery.

Clearly, to foil these impersonation attacks it suffices to apply almost any less trivial function h before the square root is taken, as long as the bits of I_A are sufficiently "mixed up" with the bits of A's public key. This is how the application of this certification scheme is fixed in [1]. However, whenever the bits of I_A can be recognized in a sufficiently short initial segment of the concatenation, these approaches will work.

3.6. *Identity Concatenated to the Right*

Let $h(I_A, N_A) = 2^m N_A + I_A$, i.e., the identity appears in the m least significant bits of the message to be signed. Also in this case we can build a forgery, if, for instance, $I_A \equiv 1 \bmod 4$. If that is the case, the polynomial $X^2 - I_A \equiv X^2 - 1 = (X - 1)(X + 1)$ in $(\mathbf{Z}/4\mathbf{Z})[X]$. Using, for instance, Hensel's lemma [7], this factorization can be lifted to a factorization in $(\mathbf{Z}/2^m\mathbf{Z})[X]$, i.e., we can efficiently find an integer r with $0 \le r < 2^m$ such that $X^2 - I_A \equiv (X - r)(X + r)$ in $(\mathbf{Z}/2^m\mathbf{Z})[X]$. Such an r can be trivially used to fabricate a forgery of length $n/2$: guess $(n/2) - m$ bit x's until the leftmost $n - m$ bits of the (n bit) square of $C_x = 2^m x + r$ represents a feasible number. Notice that the least significant m bits of $C_x^2 \bmod N$ indeed represent I_A.

With a little more work such an r can also be used to make a forgery of length $n - m - 4$. Let x have $n - 2m - 4$ bits, and form C_x as above. We require that the m least significant bits of $(2^m x + r)^2 \bmod N$ represent I_A. Because $(2^m x + r)^2 = 2^{2m} x^2 + 2^{m+1} xr + r^2$, because $2^{m+1} xr + r^2 < N/2$, and because $2^{m+1} xr + r^2 \equiv I_A \bmod 2^m$, we find that x must be chosen in such a way that $2^{2m} x^2 \bmod N$ is divisible by 2^m and $< N/2$, i.e., such that it does not "contaminate" the correct bits that are already in place at the m least significant positions. This can, for instance, be achieved by selecting x such that $x^2 \bmod N < N/2^{2m+1}$, which is satisfied if the leading $2m + 2$ bits of $x^2 \bmod N$ are zero. Using the first approach from Section 3.7 on an "I_A" consisting of $2m + 2$ zero bits, we can generate quite a few $n - 2m - 4$ bit x's that satisfy this condition, if $2m + 2 < n$. After a number of trials that depends on the protocol, this should enable us to find an x such that the $n - m$ most significant bits of the n-bit number $(2^m x + r)^2 \bmod N$ represent a feasible number. Notice that the forgery is of length $n - 2m - 4 + m = n - m - 4$. Of course, this approach extends to any case where the polynomial $X^2 - I_A$ can be factored over $(\mathbf{Z}/2^m\mathbf{Z})[X]$.

4. Conclusion

This paper illustrates that it is easy to overlook potential weaknesses in otherwise strong algorithms. For both key certification schemes discussed in this paper the security was believed to be based on the usual assumptions that discrete logarithms (ElGamal) and factoring (Rabin) are hard; proper selection of the public keys (finite field or composite number) supposedly sufficed to create an unbreakable scheme. In both instances apparently more care has to be taken to make the transition from "research scheme" to "practical application" successful and viable. We are, however, not aware of any applications that do more than the bare-bones protocol, and in the literature the need for extensions is not mentioned.

We did not address the many questions that are related to proper selection of public keys. The difficulty of discrete logarithms modulo p is probably not weakened if the factorization of $p - 1$ is made public as well (as long as there is at least a sufficiently large prime in the factorization), but this information can be used to check that the claimed generator indeed has the proper order. Thus the integrity of discrete logarithm public keys can be made believable in practical situations, if the characteristic is sufficiently large to rule out number field sieve attacks [5], [6]. Do similar methods exist to prove the integrity of composite moduli in factoring-based systems?

Acknowledgments

We gratefully acknowledge the very useful comments of two anonymous referees, one of whom pointed out references [2] and [11] to us.

References

[1] M. J. Beller, L. F. Chang, and Y. Yacobi, Privacy and authentication on a portable communications system, *IEEE GLOBECOM '91 Conference Record*, pp. 1922–1927.

[2] E. Brickell and J. DeLaurentis, *An Attack on a Signature Scheme Proposed by Okamoto and Shiraishi*, Lecture Notes in Computer Science, Vol. 218, Springer-Verlag, Berlin, pp. 28–32.

[3] CY1024 Processor Chip Key Management Applications, Cylink, 1986.

[4] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, **31** (1985), 469–472.

[5] D. M. Gordon, Discrete logarithms in $GF(p)$ using the number field sieve, *SIAM J. Discrete Math.*, **6** (1993), 124–138.

[6] D. M. Gordon, Designing and detecting trapdoors for discrete log cryptosystems, *Proc. Crypto '92*, to appear.

[7] D. E. Knuth, *The Art of Computer Programming*, Vol. II, 2nd edn., Addison-Wesley, Reading, MA, 1981.

[8] J. K. Omura, Private communication, 1991.

[9] M. O. Rabin, Digitalized Signatures and Public-Key Functions as Intractable as Factorization, TR 212, MIT Laboratory for Computer Science, 1979.

[10] D. Seeley, Password cracking: a game of wits, *Comm. ACM*, **32** (1989), 700–703.

[11] B. Vallée, M. Girault, and Ph. Toffin, *How To Break Okamoto's Cryptosystem by Reducing Lattice Bases*, Lecture Notes in Computer Science, Vol. 330, Springer-Verlag, Berlin, pp. 281–292.