A Small Subgroup Attack on Arazi's Key Agreement Protocol

Dan Brown
Certicom Research, Canada
dbrown@certicom.com

Alfred Menezes
Dept. of C&O, University of Waterloo, Canada
ajmeneze@uwaterloo.ca

Abstract

In 1993, Arazi presented a key agreement protocol that integrates the Diffie-Hellman key agreement protocol and the digital signature algorithm (DSA). In this note, we present a small subgroup attack on Arazi's protocol whereby an attacker can learn another entity's DSA private key. The attack illustrates the importance of public-key validation, i.e., checking that group elements received from another party do indeed have the prescribed order. The attack also demonstrates that extreme care must be exercised when two or more cryptographic protocols are combined to design a new protocol.

Keywords: small subgroup attack, key agreement

1 Introduction

Small subgroup attacks on Diffie-Hellman key agreement procotols [3] are described in [5]. In such attacks, a party A attempts to deduce partial information about B's private key b by inducing B to raise a group element α cleverly chosen by A to the power b. For example, if α is an element in a subgroup of small order l, then if A learns α^b she can efficiently find $b \bmod l$ by exhaustive search.

This paper presents a realistic small subgroup attack on a key agreement protocol proposed in 1993 by Arazi [2]. Arazi's protocol is interesting because it integrates the Diffie-Hellman key agreement protocol with a standardized signature scheme, the DSA [7]. The attack we present is noteworthy because it is the first example of how the partial-information

leakage attack of Howgrave-Graham and Smart [4] on ElGamal-type signature schemes can be applied to attack a protocol with a different cryptographic objective, namely key agreement. We begin by reviewing the DSA signature scheme and Arazi's key agreement protocol before presenting the attack in Section 2.

The Digital Signature Algorithm. Domain parameters for DSA are (p, q, g) where p and q are primes such that q divides $p-1$ and q has bitlength at least 160, and $g \in \mathbb{Z}_p^*$ is an element of order q. H denotes a cryptographic hash function such as SHA-1. Each entity A has a private key a selected uniformly at random from the integers in the interval $[1, q-1]$ (denoted $a \in_R [1, q-1]$) and a corresponding public key $Q_A = g^a \bmod p$.

To sign a message m, A does the following:

1. Select $x \in_R [1, q-1]$.
2. Compute $R = g^x \bmod p$ and $r = R \bmod q$. If $r = 0$ then go to Step 1.
3. Compute $x^{-1} \bmod q$ and $e = H(m)$.
4. Compute $s = x^{-1}\{e + ar\} \bmod q$. If $s = 0$ then go to Step 1.
5. A's signature for the message m is (r, s).

To verify A's signature (r, s) on m, B obtains authentic copies of the domain parameters (p, q, g) and A's public key Q_A and does the following:

1. Verify that r and s are integers in the interval $[1, q-1]$.
2. Compute $e = H(m)$, $w = s^{-1} \bmod q$, $u_1 = ew \bmod q$, and $u_2 = rw \bmod q$.
3. Compute $V = g^{u_1} Q_A^{u_2} \bmod p$ and $v = V \bmod q$.
4. Accept the signature if and only if $v = r$.

Arazi's Key Agreement Protocol. This uses the one-time public keys R in DSA as the ephemeral public keys for the Diffie-Hellman key agreement protocol. In addition, R is signed using DSA in order to authenticate the ephemeral public keys. Since attacks on protocols can be very subtle, we describe Arazi's protocol in full detail.

Domain parameters are (p, q, g) as defined for DSA. A's key pair is (a, Q_A), and B's key pair is (b, Q_B).

1. A does the following:
   (a) Select $x \in_R [1, q-1]$.
   (b) Compute $R_A = g^x \bmod p$ and $r_A = R_A \bmod q$. If $r_A = 0$ then go to 1(a).

(c) Compute $x^{-1} \bmod q$ and $e_A = H(R_A)$.

(d) Compute $s_A = x^{-1}\{e_A + ar_A\} \bmod q$. If $s_A = 0$ then go to 1(a).

(e) Send (R_A, s_A) to B.

2. B does the following:

(a) Compute $r_A = R_A \bmod q$. Verify that r_A, s_A are integers in $[1, q-1]$.

(b) Compute $e = H(R_A)$, $w = s_A^{-1} \bmod q$, $u_1 = ew \bmod q$, $u_2 = r_A w \bmod q$.

(c) Compute $V = g^{u_1} Q_A^{u_2} \bmod p$.

(d) Terminate the protocol if $V \neq R_A$.

(e) Select $y \in_R [1, q-1]$.

(f) Compute $R_B = g^y \bmod p$ and $r_B = R_B \bmod q$. If $r_B = 0$, go to 2(e).

(g) Compute $y^{-1} \bmod q$ and $e_B = H(R_B)$.

(h) Compute $s_B = y^{-1}\{e_B + br_B\} \bmod q$. If $s_B = 0$ then go to 2(e).

(i) Send (R_B, s_B) to A.

(j) Compute the shared secret $K = R_A^y \bmod p$.

3. A does the following:

(a) Compute $r_B = R_B \bmod q$. Verify that r_B, s_B are integers in $[1, q-1]$.

(b) Compute $e = H(R_B)$, $w = s_B^{-1} \bmod q$, $u_1 = ew \bmod q$, $u_2 = r_B w \bmod q$.

(c) Compute $V = g^{u_1} Q_B^{u_2} \bmod p$.

(d) Terminate the protocol if $V \neq R_B$.

(e) Compute the shared secret $K = R_B^x \bmod p$.

Remark. An important attribute of ephemeral Diffie-Hellman key agreement is *known-key security*. That is, if an adversary C learns one or more secrets K established by A and B, then this does not help the adversary learn any other secrets established by A and B. Nyberg and Rueppel [9] showed that Arazi's protocol does not have the attribute of known-key security. Suppose that (R_A, s_A) and (R_B, s_B) are the (non-secret) messages exchanged in one run of Arazi's protocol. Let $r_A{=}R_A \bmod q$, $w{=}s_A^{-1} \bmod q$, $u_1{=}H(R_A)w \bmod q$ and $u_2{=}r_A w \bmod q$. Similarly, let $r_B{=}R_B \bmod q$, $\overline{w}{=}s_B^{-1} \bmod q$, $\overline{u}_1{=}H(R_B)\overline{w} \bmod q$ and $\overline{u}_2{=}r_B\overline{w} \bmod q$. Then

$$K = g^{xy} = g^{(u_1+au_2)(\overline{u}_1+b\overline{u}_2)} = g^{u_1\overline{u}_1} Q_A^{\overline{u}_1 u_2} Q_B^{u_1\overline{u}_2} (g^{ab})^{u_2\overline{u}_2}.$$

This shows that shared secrets can be computed from the long-term shared secret g^{ab} and other non-secret information. Hence if C learns K, then C can efficiently compute g^{ab}, and thus compute all other secrets established by A and B.

Remark. In order to save bandwidth, Arazi's protocol can be modified so that (r_A, s_A) and (r_B, s_B) are exchanged instead of (R_A, s_A) and (R_B, s_B). The attack presented in this paper also works on this modified protocol.

2 The Attack

The goal of the attack is for A to obtain B's DSA signing key b. In practice, the DSA prime p has bitlength 1024, while q has bitlength 160. Let t be the highest power of 2 that divides $p-1$. Certainly we have $t \geq 1$. We henceforth assume without much loss of generality that $t \geq 2$. Let $\alpha \in \mathbb{Z}_p^*$ be an element of order 2^t. The attack proceeds as follows.

Public key selection. A selects a public key $Q_A = g^a \alpha \bmod p$. Note that Q_A is an *invalid* public key since $Q_A^q \not\equiv 1 \pmod{p}$. A then gets Q_A certified by a Certification Authority (CA)—the certificate on Q_A produced by the CA can be used by B to verify that she has an authentic copy of A's public key. In practice (as dictated by PKI standards such as [1, Section 2.3] and [6, Section 4]), the CA does not perform public key validation, i.e., the check that $Q_A^q \equiv 1 \pmod{p}$ and $Q_A \not\equiv 1 \pmod{p}$. Rather, the CA performs a proof of possession (POP) of a private key test whereby A has to submit a DSA signature generated with respect to Q_A on some message m of a predetermined format, and the signature is thereafter verified by the CA. The following shows that A can successfully pass the POP test even though Q_A is invalid. A repeatedly generates DSA signatures (r, s) on m using private key a until $u_2 \equiv 0 \pmod{2^t}$, where $u_2 = rs^{-1} \bmod q$; heuristically, the expected number of trials is 2^t. The CA will then accept (r, s) as A's signature for m with public key Q_A since

$$Q_A^{u_2} \equiv (g^a)^{u_2} \alpha^{u_2} \equiv (g^a)^{u_2} \pmod{p}.$$

Obtaining partial information about B's ephemeral private keys y. A now repeatedly selects $x \in_R [1, q-1]$ and computes $R_A = g^x \alpha \bmod p$, $r_A = R_A \bmod q$, $e_A = H(R_A)$ and

$$s_A = x^{-1}\{e_A + a r_A\} \bmod q, \tag{1}$$

until $r_A \neq 0$, $s_A \neq 0$, and

$$(r_A s_A^{-1} \bmod q) \equiv 1 \pmod{2^t}. \tag{2}$$

We can reasonably assume that $r_A s_A^{-1} \bmod q$ is uniformly distributed among the equivalence classes of integers modulo 2^t for x selected uniformly at random from $[1, q-1]$; then the expected number of trials is 2^t. A now sends (R_A, s_A) to B who accepts this pair (Steps 2(a)–2(d) of the Arazi protocol) since

$$V \equiv g^{u_1} Q_A^{u_2} \equiv g^{u_1} (g^a)^{u_2} \alpha^{u_2} \equiv g^x \alpha \pmod{p}$$

by (1) and (2). B then sends R_B and s_B to A. B computes $K = R_A^y \bmod p$; note that $K \equiv g^{xy} \alpha^y \pmod{p}$. A computes $K' = R_B^x \bmod p$; note that $K' \equiv g^{xy} \pmod{p}$.

Now, suppose that B uses K (or a key derived from it, for example by taking the appropriate number of bits of $H(K)$) in a symmetric-key protocol with A. For concreteness, suppose that B uses K as a key for a secure MAC algorithm and sends A the tag $T = \mathrm{MAC}_K(M)$ for some message M. A computes $T_i = \mathrm{MAC}_{K_i}(M)$ for each $i \in [0, 2^t - 1]$ where $K_i = K' \alpha^i \bmod p$ until $T_i = T$. When this occurs, then with overwhelming probability (since the MAC is secure) $K_i = K$ whence $i = y \bmod 2^t$. In this way, A learns the t least significant bits of B's ephemeral private key y.

Computing B's private key b. By repeating the procedure described above d times, A obtains d messages and B's DSA-signatures on them, as well as the t least significant bits of each ephemeral private key. The technique introduced by Howgrave-Graham and Smart [4] (see also [8]) can then be used to efficiently recover B's private key b.

Experiments conducted by Nguyen and Shparlinski [8] were successful in recovering b when $(t, d) = (4, 70)$ and $(t, d) = (3, 100)$. It failed when $(t, d) = (2, 150)$. Thus, assuming that $t \geq 3$, our attack can be expected to be successful for some $d \leq 70$. This number of iterations d is very reasonable in many scenarios, for example when A is a client and B is a web server. Thus our attack on Arazi's protocol should be considered a realistic one.

3 Conclusions

We have presented a realistic attack on a key agreement protocol proposed by Arazi. The attack can be prevented by checking that the long-term public keys Q_A or ephemeral public keys R_A are indeed non-trivial elements of the unique subgroup of order q of $\mathbb{Z}_p^*$. (However, we do not claim that other attacks are not possible even with these checks in place.) The attack illustrates the importance of public-key validation, i.e., checking that group elements received from another party do indeed have the prescribed order. The attack also demonstrates that extreme care must be exercised when two or more cryptographic protocols are combined to design a new protocol.

Acknowledgements

We thank Colin Boyd for bringing reference [9] to our attention.

References

[1] C. Adams and S. Farrell, *Internet X.509 Public Key Infrastructure: Certificate Management Protocols*, RFC 2510, March 1999. Available from http://www.ietf.org

[2] B. Arazi, "Integrating a key distribution procedure into the digital signature standard", *Electronics Letters*, **29** (1993), 966-967.

[3] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, **22** (1976), 644-654.

[4] N. Howgrave-Graham and N. Smart, "Lattice attacks on digital signature schemes", *Designs, Codes and Cryptography*, **23** (2001), 283-290.

[5] C. Lim and P. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup", *Advances in Cryptology–Crypto '97*, Lecture Notes in Computer Science, **1294** (1997), 249-263.

[6] M. Myers, C. Adams, D. Solo and D. Kemp, *Internet X.509 Certificate Request Message Format*, RFC 2511, March 1999. Available from http://www.ietf.org

[7] National Institute of Standards and Technology, *Digital Signature Standard*, FIPS Publication 186, 1994.

[8] P. Nguyen and I. Shparlinski, "The insecurity of the Digital Signature Algorithm with partially known nonces", *Journal of Cryptology*, **15** (2002), 151-176.

[9] K. Nyberg and R. Rueppel, "Weaknesses in some recent key agreement protocols", *Electronics Letters*, **30** (1994), 26-27.