Making Use of the Subliminal Channel in DSA

Seth Hardy

shardy@aculei.net
http://www.aculei.net/∼shardy

The Fifth HOPE
July 10, 2004

Outline

1. Digital Signatures
   - What they are and what they do
   - ElGamal signatures
   - DSA signatures
2. Subliminal Channels
   - Overview
   - The subliminal channel in DSA
   - Malicious use of the subliminal channel
   - A demonstration using `subdsakey.pl`
3. Summary and Questions

Part I

Digital Signatures

What is a Digital Signature?

- A digital signature is a number (or numbers) dependent on some secret known to the signer, and on the signed message.
- Digital signatures are implemented using public-key cryptography: the signer has a *private key* used for creating signatures, and a *public key* used for signature verification.

What Digital Signatures Do

Digital signatures should emulate the functionality of a handwritten signature, including:

- Authentication
- Verification of data integrity
- Non-repudiation of signatures

ElGamal Signatures

- The ElGamal algorithm can be used for both encryption and digital signatures, although in practice it is generally not used for digital signatures.
- DSA is a variant of the ElGamal signature scheme.
- ElGamal signatures have a subliminal channel that is not as readily usable as the one in DSA.

ElGamal Key Generation

ElGamal key pairs are generated in the following way:

- Let p be a large prime, and α a generator of $\mathbb{Z}_p^*$.
- Randomly choose an integer x such that $1 \leq x \leq p - 2$.
- Compute $y = \alpha^x \pmod{p}$.
- Let the public key be (p, α, y) and the private key be x.

ElGamal Signatures

An ElGamal signature on a message m is computed as follows:

- Select a random k s.t. $1 \leq k \leq p - 2$ and $\gcd(k, p - 1) = 1$.
- Compute $r = \alpha^k \pmod{p}$.
- Compute $k^{-1} \pmod{p - 1}$.
- Compute $s = k^{-1}(h - xr) \pmod{p - 1}$.
- The signature for m is (r, s).

ElGamal Signature Verification

An ElGamal signature (r, s) on a message m is verified as follows:

- Check that $1 \leq r \leq p - 1$; if not, reject the signature.
- Compute $v_1 = y^r r^s \pmod{p}$.
- Compute $v_2 = \alpha^h \pmod{p}$.
- Accept the signature iff $v_1 = v_2$.

The Digital Signature Standard: FIPS 186-2

- The Digital Signature Algorithm is defined as part of the Digital Signature Standard (DSS) in NIST FIPS 186.
- The original standard was FIPS 186, published in May 1994. The standard was updated in FIPS 186-1 (December 1998) and FIPS 186-2 (January 2000).
- The FIPS PUBS can be found at the Computer Security Resource Center of the National Institute of Standards and Technology: http://csrc.nist.gov

DSA Key Generation

DSA key pairs are generated in the following way:

- Let q be a prime number s.t. $2^{159} < q < 2^{160}$.
- Choose t s.t. $0 \leq t \leq 8$.
- Let p be a prime number where $2^{511+64t} < p < 2^{512+64t}$ s.t. $q | (p-1)$.
- Let α be a generator of the cyclic group of order q in $\mathbb{Z}_p^*$.
- Randomly choose an integer x s.t. $1 \leq x \leq q-1$.
- Compute $y = \alpha^x \pmod{p}$.
- Let the public key be (p, q, α, y) and the private key be x.

DSA Signing

A DSA signature on a message m is computed as follows:

- Select a random k s.t. $0 < k < q$.
- Compute $r = (\alpha^k \bmod p) \bmod q$.
- Compute $s = k^{-1}(h + xr) \pmod{q}$, where h is the SHA-1 hash of m.
- The signature for m is (r, s).

DSA Verification

A DSA signature (r, s) on a message m is verified as follows:

- Check that $0 < r < q$ and $0 < s < q$; if not, reject the signature.
- Compute $u_1 = s^{-1}h \pmod{q}$.
- Compute $u_2 = rs^{-1} \pmod{q}$.
- Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$.
- Accept the signature iff $v = r$.

Part II

Subliminal Channels

What is a Subliminal Channel?

A *subliminal channel* is a way of embedding information in public communication in an undetectable way. Some sort of shared secret (a key, knowledge of what to look for) is needed to reconstruct the subliminal information.

Subliminal Channel Types

There are two types of subliminal channels in digital signatures:

- *Broadband* channels use all available "space" in the signature (i.e. bits that aren't being used for preventing forgery, alteration, or transplantation).
- *Narrowband* channels use a fraction of the available "space" but generally offer some other advantage over broadband channels.

The Subliminal Channels in DSA

- 1985: Gustavus Simmons found a subliminal channel in ElGamal signatures:
  - The recipient needs to know the private key.
  - Only a fraction of possible messages can actually be sent.
- 1991: DSS (including DSA) is proposed.
- 1993: Simmons finds one broadband and two narrowband subliminal channels in DSA:
  - The broadband channel requires sharing the private key, but can send all possible messages.
  - The narrowband channels transmit much less information, but do not require sharing the private key.
- 1994: DSS is accepted as the first government digital signature standard.

Sending a Broadband Subliminal Message via DSA

The objective is to send a message embedded in a digital signature on any message, such that:

- The digital signature is still valid.
- There is no way to prove that the message is there without some shared secret.
- There is no way to reconstruct the message without the shared secret.

Here we will look at the broadband channel in DSA.

How it Works

The math is fairly simple: if you know the private key, k is the only unknown value. This means it can be solved for in a straightforward manner:

$$\begin{aligned} s &= k^{-1}(h - xr) \bmod q \\ sk &= h - xr \bmod q \\ k &= s^{-1}(h - xr) \bmod q \end{aligned}$$

To embed a subliminal message using the broadband channel, all you need to do is set k to a specific value instead of choosing it randomly.

The Simple Idea

The subliminal channel works the other way around, too:

$$\begin{aligned} s &= k^{-1}(h - xr) \bmod q \\ sk &= h - xr \bmod q \\ xr &= sk - h \bmod q \\ x &= r^{-1}(sk - h) \bmod q \end{aligned}$$

So, if you know the value of k, you can use the subliminal channel "backwards" to compute the private key x. In this case, the private key is the subliminal message!

GPG Patch

```
--- dsa.c       2004-06-12 20:45:42.675357368 -0400
+++ dsa-patched.c       2004-06-12 20:45:53.315739784 -0400
@@ -286,7 +286,8 @@
MPI tmp;

/* select a random k with 0 < k < q */
-    k = gen_k( skey->q );
+    k = mpi_alloc_secure( mpi_get_nlimbs(skey->q) );
+    mpi_fdiv_r( k, hash, skey->q );

/* r = (a^k mod p) mod q */
mpi_powm( r, skey->g, k, skey->p );
```

Reconstructing a Leaked OpenPGP Key

Reconstructing a leaked OpenPGP key is easy!

- Solve for x as described above.
- Create the private key: x and p, q, α, y from the public key.
- Encapsulate the private key in a certificate.
- Create a keyblock and add the secret key certificate.
- Copy the userid from the public key and add it to the keyblock.
- Self-sign the public key certificate and add it to the keyblock.
- Optionally armor the keyblock, and write it to a file.

Proof of concept: `subdsakey.pl` does it all!

Part III

Summary and Conclusions

Summary

- As Simmons said, subliminal communication is *easy* using DSA.
- The subliminal channels appear to be a feature, not a bug.
- Subliminal communication can be used for all kinds of communication, some of which are extremely malicious.
- **Always verify the trustworthiness of your encryption program if you don't want to risk leaking your keys!**

Future Work

Currently, I am working on:

- A program in Perl to embed/extract subliminal messages from signatures.
- A patch to GnuPG to provide this same functionality directly.
- A program and a patch to utilize the narrowband subliminal channels.
- Other assorted utilities which will automatically scan signatures for subliminally leaked keys.

References

[0] Fips 186–2, Digital Signature Standard, Federal Information Processing Standards Publication 186–2, US Dept. of Commerce/NIST National Institute of Standards and Technology, 1994.

[1] Gustavus J. Simmons. Subliminal communication is easy using the DSA. Lecture Notes in Computer Science, 765:218–232, 1994.

Questions?