On *Key*-collisions in (EC)DSA Schemes

(CRYPTO 2002 - Rump Session Note)

Tomáš Rosa, tomas.rosa@i.cz

[1] ICZ, Prague, Czech Republic

[2] Dept. of Computer Science and Eng., FEE, Czech Technical University in Prague

We start by reviewing one of those practical definitions of a non-repudiation service: *The third independent party can be convinced that a particular event did (or did not) occur* ([3]). Next we continue by the definition of what we mean by the term *k*-collision: *Let (m, S) be a message and its signature. The pair of public keys (Pub_A, Pub_B), $Pub_A \neq Pub_B$, is said to be a key-collision (k-collision) if there is (m, S), such that S is a valid signature of m under both public keys Pub_A and Pub_B. Such a signature S we refer to as a k-colliding signature.*

Now we briefly show some scenarios, under which an ability to find a particular *k*-collision may lead to attacks on a non-repudiation service. In the first scenario we assume that there are two companies (represented as users *A*, *B*), which want to enter a tender. But actually only one of them will want to take this contract later on. We assume here it to be illegal for a selected company to pass its contract to another one, once this selected company realizes that it actually doesn't want that work under given (unpredictable) conditions. That is the exact case, where *k*-collisions may help. Our two companies may agree to find a *k*-collision for their particular submission message. Later on, when the selection is made, actually only one of them will apply its signature. It is also reasonable to assume that during a tender evaluation there will be now identities of submitting companies available, so it may be possible to do this change even without any suspicion. Comparing to classical attacks on non-repudiation, this attack actually leads to the claim: "It has been someone else, who has signed this message…" instead of the claim: "I have not signed this message, I have signed that one…"

Another attack scenario may arise if it is possible to compute the second public key Pub_B in the *k*-collision without any cooperation with the owner of the public key Pub_A. We call such a *k*-collision as a non-cooperatively computable. Let us assume that the owner of the public key Pub_A had signed some message in a past and that she/he had also timestamped it. This is a possible scenario, when the user *A* wants to prove its authorship of her/his document later on. Let us assume that the time stamp does not cover the value of Pub_A and that there was such a signature scheme used, which allows *k*-collisions to be non-cooperatively computed. In this case an attacker (as the user *B*) may be able to compute its own public key Pub_B, which forms a *k*-collision with Pub_A on this given message and its signature. In this way an attacker in fact steals *A*'s signature and perhaps also her/his authorship. The claim, which this attack leads to, is: "It has been me, who has signed this message, not her/him…"

In the following text we are going to show that in DSA scheme ([1]) it is trivially possible to find a *k*-collision for an arbitrary chosen (*m*, *S*) and that this *k*-collision is non-cooperatively computable.

Definition: DSA instance. *The DSA instance consists of the public parameters (p, q, g), the public key y and the private key x.*

Definition: Correct DSA instance. *The DSA instance is said to be correct here if:*

p, q are both primes,
$2^{L-1} < p < 2^L$, *for some L,* $L = 512 + 64j$, *where j is an integer,* $0 \leq j \leq 8$,
$2^{159} < q < 2^{160}$, $q \mid (p-1)$,
$g^q \bmod p = 1$,
$g^x \bmod p = y$.

Let us have (*m*, *S*), where *S*, $S = (r, s)$, is a valid DSA signature of *m* with respect to the public key Pub_A, $Pub_A = y_A$, together with the public parameters (p_A, q_A, g_A). What we want to do is to search for the second public key Pub_B, $Pub_B = y_B$, together with the public parameters (p_B, q_B, g_B) and the private key x_B, such that:

- the DSA instance (p_B, q_B, g_B, y_B, x_B) is correct,
- the public keys y_A and y_B form the *k*-collision on the *k*-colliding signature *S* and the message *m*.

Algorithm: Computing a DSA *k*-collision. Let us denote h = SHA-1(*m*). We start by setting $p_B = p_A = p$, $q_B = q_A = q$. Then we continue as follows:

i) Compute: $\alpha = g_A^{wh} y_A^{wr} \bmod p$, where $w*s \equiv 1 \pmod q$
- Note that $\alpha = g_A^{kA} \bmod p$, where k_A is the unknown DSA ephemeral message key for (m, S). Also note that $\alpha \bmod q = r$.

ii) Choose an integer k_B, $1 < k_B < q$, and compute z as $z*k_B \equiv 1 \pmod q$
- Note that the attacker should keep the values of k_B and z secret, eventually she/he may want to discard them when the whole algorithm is done.

iii) Set $\boldsymbol{g_B} = \alpha^z \bmod p$
- Note that $g_B^{kB} \equiv \alpha \pmod p$, $(g_B^{kB} \bmod p) \bmod q = r$ and that $g_B^q \equiv 1 \pmod p$.
- The k_B becomes the DSA ephemeral message key for the user B.
- We may also test whether $g_B \overset{?}{=} g_A$. In such an unlikely case it holds that $k_B \equiv k_A \pmod q$, so the user B knows A's message key k_A and the value of the private key x_A can be computed easily.

iv) Finally we compute:
- $\boldsymbol{x_B} = t(k_B s - h) \bmod q$, where $r*t \equiv 1 \pmod q$
- $\boldsymbol{y_B} = g_B^{xB} \bmod p$
- Note that now $s = (h + rx_B)z \bmod q$ and $r = (g_B^{kB} \bmod p) \bmod q$.
- Also note that $x_B = ht(\beta^{-1} - 1) + \beta^{-1} x_A \bmod q$ and $y_B = g_A^{ht(1-\beta)} y_A \bmod p$, where $\beta \equiv k_A z \pmod q$, $\beta*\beta^{-1} \equiv 1 \pmod q$.

It can be easily shown that the DSA instance $(p_B, q_B, g_B, y_B, x_B)$ is correct and that the public keys (y_A, y_B) form a k-collision on the k-colliding signature S (with the high probability). Note that the user A cannot determine x_B, while the user B cannot determine x_A (both hold under the assumption that DSA is unbreakable). For a sketch of proofs see notes commenting the algorithm above. Also there are no obvious marks of an attack: It is allowed to share public parameters among many (EC)DSA instances (sometimes it is even recommended) and except this partial sharing, there is nothing suspicious about the DSA instance of user B. Because the algorithm described above uses those algebraic properties, which are common for both DSA and ECDSA ([2]), it can be trivially extended on ECDSA too. Finally we note that the algorithm uses the same operations as those which are used during (EC)DSA signing/verification. Therefore this attack is feasible whenever general (EC)DSA usage is feasible.

Summary

The notion of *k*-collisions was introduced and possible attack scenarios based on this notion were discussed briefly. Then an algorithm for a non-cooperatively computable *k*-collision finding was presented for (EC)DSA. From here follows that (EC)DSA suffers from attacks based on these *k*-collisions. The main problem behind that is the lack of restrictions on the generation of the subgroup generator *g*. Due to that it is possible to make a so called *partial inversion of the DSA instance generation process*. If there were a mechanism for a certification of *g*, similar to those ones used for values *p*, *q* ([1]), then this attack would not be possible. In addition to the article [5] this is another motivation to change the standard [1] in that part. The same applies also for ECDSA analogically.

As a temporary countermeasure we may suggest to include detailed public key information into the data to be signed. For instance in PKCS formats ([4]) this option already exists (c.f. `authenticatedAttributes` field in [4, PKCS#7]), but it has been used in a practice very seldom. However we note that it can still be broken by a 2nd order *k*-collision attack, which can be illustrated by the following properties: *different message*, *different public keys* and *the same signature*. These kinds of k-collisions are also trivially non-cooperatively computable in (EC)DSA schemes. Therefore this countermeasure must be done with strong respect to a particular PKI environment, which the (EC)DSA scheme will be applied in.

References

[1] FIPS PUB 186-2: *Digital Signature Standard (DSS)*, National Institute of Standards and Technology, January 27, 2000, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf, update: October 5, 2001.

[2] Johnson, D., Menezes, A. and Vanstone, S.: The Elliptic Curve Digital Signature Algorithm (ECDSA), *International Journal of Information Security*, Vol 1, Issue 1, pp. 36-63, Springer-Verlag, 2001.

[3] Landwehr, C.-E.: Computer Security, *International Journal of Information Security*, Vol 1, Issue 1, pp. 3-13, Springer-Verlag, 2001.

[4] *Public-Key Cryptography Standards (PKCS)*, RSA Security, available at http://www.rsasecurity.com/rsalabs/pkcs/index.html.

[5] Vaudenay, S.: Hidden Collisions on DSS, *in Proc. of CRYPTO '96*, pp. 83-88, Springer-Verlag, 1996.