6260: Applied Cryptography

Homework 6

Lecturer: Sasha Boldyreva

The skills practiced: basic usage of cryptographic libraries and brute-force attacks.

Digital Signature Algorithm (DSA) Key-Recovery from Nonce

Let's recall DSA signature.

- There are public parameters (p, q, g), where p and q are large primes; $p-1$ is a multiple of q and g is a group generator. Also, $H(\cdot)$ is a cryptographic hash function.
- Key generation KeyGen generates secret key $x \xleftarrow{\$} \mathbb{Z}_q^*$, and public key $y \leftarrow g^x \bmod p$, and outputs (x, y).
- Signing $\mathsf{Sign}(m)$:
  - generate a random nonce $k \xleftarrow{\$} \mathbb{Z}_q^*$.
  - $r \leftarrow \left(g^k \bmod p\right) \bmod q$.
  - $s \leftarrow (k^{-1}(H(m) + xr)) \bmod q$.
  - output pair (r, s).

What is the vulnerability?

It is possible that the range over which the random nonce k is selected is very small. If an attacker wants to retrieve the private key from the given signature (r, s) and the message m, he can exploit the fact that nonce k is generated over a small range.

How does the attack work?

The attacker has access to the message m and (r, s) pair. He can first try to recover k by brute-force the range of k (assuming it is small). Then with k to recover the secrete key x from s.

Task

Assume instead of using the large set $\mathbb{Z}_q^*$, random nonce k is selected randomly from a small set $\{1, 2, \ldots, 2^{16} - 1\}$. You are provided with an `input.json` file containing:

- Public parameters: (p, q, g), where $|q| = 160$, $|p| = 1024$; for simplicity we instantiate $H(\cdot)$ with SHA-1.
- Public key $y := g^x \bmod p$
- Message m and its signature pair (r, s) signed with x and k
- Hash $h = \text{SHA-1}(m)$ in hexadecimal representation

You are expected to compute k and produce the private key x that was used to sign the message m. There is no restriction on the language nor external cryptographic library (e.g., Crypto, OpenSSL) used for your attack. Note that we run key generation algorithm independently for each student.

Collaboration & Resources

No collaboration is allowed for this homework. You are allowed to look at the examples of how to use the API of the language and cryptographic libraries and json parsing on the internet.

Submission

Please submit following **three** files (separately, **not** in a zip file),

- A textfile `report.txt`:
  (First line) `[k]`
  (Second line) `[x]`

  Note: replace `[k]` and `[x]` with your input. Here is an example:

  ```
  29238
  597328809243624339460444057943791576827045003 84
  ```

- Your implementation source file: `source.[*]`.
  Note: replace `[*]` with your file extension, e.g., `c`, `py`.

- A textfile `README.txt`:
  List any external cryptographic library or json parsing library used in your source file.

Penalties

- For file names with wrong formatting, we will deduct 2 points each.
- For wrong formatting of `report.txt`, we will deduct 5 points.

References

[1] `https://en.wikipedia.org/wiki/Digital_Signature_Algorithm`

[2] NIST.FIPS.186-4 (latest) `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`