Introduction to Cryptography

TEK 4500 (Fall 2020)
Problem Set 11

Problem 1.

Read Chapter 12 in [BR] (Section 12.3.6 can be skipped) and Chapter 10 in [PP] (Section 10.3 can be skipped).

Problem 2.

Given an instance of the Textbook RSA signature scheme (Fig. 1) with public verification key $vk = (e, n) = (131, 9797)$, which of the following signatures are valid?

a) $(M, \sigma) = (123, 6292)$

b) $(M, \sigma) = (4333, 4768)$

c) $(M\sigma) = (4333, 1424)$

Problem 3.

Given the same Textbook RSA instance as in Problem 2, make a forgery on the message $M = 1234$. Suppose you're in the UF-CMA setting, i.e., you have access to a signing oracle that returns signatures on messages of your choice.

Problem 4.

The ECDSA signature scheme is shown in Fig. 2. It is defined over an elliptic curve group $(E(\mathbf{F}_p), +)$ having prime order q and using some generator element G. In particular, note that the multiplication on Line 4 of the signing algorithm is actually a group exponentiation in the elliptic curve group $(E(\mathbf{F}_p), +)$, i.e.,

$$kG = \overbrace{G + G + \ldots + G}^{k \text{ times}}$$

where "+" is the elliptic curve group operation (ref Problem set 9). The resulting point P has coordinates (x, y) (unless it equals the identity element $\mathcal{O}$). Similarly, the operation

RSA.KeyGen:

1: $p, q \xleftarrow{\$}$ two large prime numbers
2: $n \leftarrow p \cdot q$
3: $\phi(n) = (p-1) \cdot (q-1)$
4: **choose** $e \in \mathbf{Z}^*_{\phi(n)}$
5: $d \leftarrow e^{-1} \pmod{\phi(n)}$
6: $sk \leftarrow (d, n)$
7: $vk \leftarrow (e, n)$
8: **return** (sk, vk)

RSA.Sign(sk, M):

1: Parse sk as (d, n)
2: $\sigma \leftarrow M^d \pmod{n}$
3: **return** σ

RSA.Vrfy(vk, M, σ):

1: Parse vk as (e, n)
2: **if** $\sigma^e = M \pmod{n}$**:**
3:     **return** 1
4: **else**
5:     **return** 0

Figure 1: The Textbook RSA signature scheme.

ECDSA.KeyGen:

1: $d \xleftarrow{\$} \{1, \ldots, q-1\}$
2: $Q \leftarrow dG$
3: **return** (d, Q)

ECDSA.Sign(sk, M):

1: Parse sk as d
2: $z \leftarrow H(M)$
3: $k \xleftarrow{\$} \{1, \ldots, q-1\}$
4: $P = (x, y) \leftarrow kG$
5: $r \leftarrow x \pmod{q}$
6: **if** $r = 0$**:** go to Line 3
7: $s \leftarrow k^{-1}(z + rd) \pmod{q}$
8: **return** (r, s)

ECDSA.Vrfy(vk, M, σ):

1: Parse vk as Q
2: Parse σ as (r, s)
3: **if** $r, s \notin \{1, \ldots, q-1\}$**:**
4:     **return** 0
5: $z \leftarrow H(M)$
6: $a \leftarrow zs^{-1} \pmod{q}$
7: $b \leftarrow rs^{-1} \pmod{q}$
8: $P = (x, y) \leftarrow aG + bQ$
9: **if** $P = \mathcal{O}$**:**
10:     **return** 0
11: **if** $r = x \pmod{q}$**:**
12:     **return** 1
13: **else**
14:     **return** 0

Figure 2: The ECDSA signature algorithm parameterized on an elliptic curve group $(E(\mathbf{F}_p), +)$ having prime order $q = |E(\mathbf{F}_p)|$, a generator G, and a hash function $H : \{0,1\}^* \rightarrow \mathbf{Z}_q$.

on Line 8 of the verification algorithm is two group exponentiations of the points G (the group generator) and Q (the public key) together with one group operation, i.e.,

$$aG + bQ = \overbrace{G + \ldots + G}^{a \text{ times}} + \overbrace{Q + \ldots + Q}^{b \text{ times}}.$$

a) Show that ECDSA is a correct signature scheme, i.e., that $\mathsf{Vrfy}(vk, M, \mathsf{Sign}(sk, M)) = 1$.

b) ECDSA shares the following sharp edge with the Schnorr signature scheme (ref. Lecture 11, Slide 20): if the same k value is ever used to sign two different messages, then an attacker can obtain the private signing key d. Show this.

c) A common reason why a k value could ever happen to be used again is if the randomness source of the computer is bad. In this case the probability of picking a specific k value at Line 3 of ECDSA.Sign could be much higher than $1/q$. Given the catastrophic failure mode of ECDSA on k reuse, it would be good if we didn't have to rely on any randomness at all. And it turns out that's possible! This is called deterministic ECDSA and works as follows. On Line 3 of the Sign algorithm, instead of picking k at random, we instead derive it as

```
2: ...
3: k ← H(sk, M)
4: ...
```

where $sk = d$ is the long-term private signing key of ECDSA, M is the message to be signed, and H is a hash function. Explain why this solves the problem of k reuse.

d) Unfortunately, it turns out that making ECDSA deterministic makes it more vulnerable to certain side-channel attacks that are able to measure the power drawn while ECDSA is computing. Suggest a way of bringing non-determinism back to deterministic ECDSA, but without re-introducing the k-reuse problem.

Problem 5.

Let E denote the elliptic curve $y^2 = x^3 + x + 26 \pmod{127}$. It can be shown that $|E(\mathbf{F}_p)| = 131$, which is a prime number. Therefore any non-identity element in $E(\mathbf{F}_p)$ is a generator for the group $(E(\mathbf{F}_p), +)$. Suppose ECDSA is implemented in E, with $G = (2, 6)$ and $d = 54$.

a) Compute the public key $Q = dG$.

b) Compute the signature on a message M if we assume $H(M) = 10$ and $k = 75$.

c) Show the computations used to verify the signature constructed in part (b).

References

[BR] Mihir Bellare and Phillip Rogaway. *Introduction to Modern Cryptography*. https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf.

[PP] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.