Exercise Sheet 12

Prof. Kenny Paterson, F. Günther
M. Backendal, M. Filić

Discussion Questions (Key generation).

(a) How much entropy does the PIN for your bank card have? Can you derive secure cryptographic keys from it?

Suggested focus. Attempt these problems
before class: Problem 3,
in class: Problem 1 and 2.

Suggested reading. Reading the following sections in the Boneh-Shoup book [2] might help with the problems on this exercise sheet: Section 13.6 (PKCS1 signatures and Bleichenbacher's attack), Section 13.8 (Certificates and public-key infrastructure) and Sections 15.1-15.2 (Elliptic curve cryptography).

Problem 1 (Elliptic curve point addition). Recall that addition of points P, Q on an elliptic curve $E\colon y^2 = x^3 + a \cdot x + b$ is defined as follows in the lectures. Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$. The coordinates of $S = P + Q$ are given by (expressed as $S = (x_S, y_S)$ whenever possible)

i) **Case 1:** $x_P \neq x_Q$.

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P}, \qquad x_S = \lambda^2 - x_P - x_Q, \qquad y_S = \lambda \cdot (x_P - x_S) - y_P.$$

ii) **Case 2:** $P = Q$. (For simplicity, let $x = x_P = x_Q$ and $y = y_P = y_Q$).

$$\lambda = \frac{3x^2 + a}{2 \cdot y}, \qquad x_S = \lambda^2 - 2 \cdot x, \qquad y_S = \lambda \cdot (x - x_S) - y.$$

iii) **Case 3:** $P = -Q$.

$$S = O.$$

Prove that **Case 1** and **Case 2** can be unified into one formula as follows:

$$\lambda = \frac{x_P^2 + x_P x_Q + x_Q^2 + a}{y_P + y_Q}, \qquad x_S = \lambda^2 - x_P - x_Q, \qquad y_S = -(\lambda \cdot (x_S - x_P) + y_P).$$

Unifying the above cases in this way can help against side channel attacks.

Problem 2 (CurveBall vulnerability). Let $k \in \mathbb{N}$ be some symmetric key length. Let $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{SE}}, \mathrm{Dec}_{\mathrm{SE}})$ be an AE-secure symmetric encryption scheme defined for key space $\mathcal{K} = \{0,1\}^k$. Let $H\colon (\{0,1\}^*)^3 \to \{0,1\}^k$ be a hash function. Below we define a public-key encryption scheme $\mathsf{PKE} = (\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ that implements ECIES (i.e., DHIES in the ECC setting), as seen in the lectures. We modify the standard PKE syntax to include *public parameters* pp as an additional input argument to each of the constituent PKE algorithms. Public parametrs $pp = (p, a, b, P, q)$ define the elliptic curve $E\colon y^2 = x^3 + a \cdot x + b$ over $\mathbb{F}_p$ (for a large prime $p \in \mathbb{N}$) with non-O base point $P \in E(\mathbb{F}_p)$ of (prime) order q (meaning $[q]P = O$).

$\underline{\mathrm{KGen}(pp)}$	$\underline{\mathrm{Enc}(pp, pk, m)}$	$\underline{\mathrm{Dec}(pp, sk, c)}$
$(p, a, b, P, q) \leftarrow pp$	$(p, a, b, P, q) \leftarrow pp$; $X \leftarrow pk$	$(p, a, b, P, q) \leftarrow pp$; $(x, X) \leftarrow sk$
$x \leftarrow\!\!\$\ \mathbb{Z}_q$; $X \leftarrow [x]P$	$r \leftarrow\!\!\$\ \mathbb{Z}_q$; $Y \leftarrow [r]P$; $Z \leftarrow [r]X$	$(Y, c_{\mathrm{SE}}) \leftarrow c$; $Z \leftarrow [x]Y$
$pk \leftarrow X$; $sk \leftarrow (x, X)$	$k \leftarrow H(Z, X, Y)$	If $Y \notin E(\mathbb{F}_p)$ or $[q]Y \neq O$:
Return (pk, sk)	$c_{\mathrm{SE}} \leftarrow\!\!\$\ \mathrm{Enc}_{\mathrm{SE}}(k, m)$	Return `Fail`
	Return (Y, c_{SE})	$k \leftarrow H(Z, X, Y)$
		$m \leftarrow \mathrm{Dec}_{\mathrm{SE}}(k, c_{\mathrm{SE}})$
		`Return` m

Consider an implementation of the above that works as follows. The first time Goofy wants to send a message to Daisy, he goes to Daisy's website and gets the public key pk and the public parameters pp used by her, along with a Certificate Authority's (CA) signature that binds pk, pp to Daisy's identity. Goofy then checks the CA signature to verify the authenticity of this information. If everything passes, then Goofy uses pk, pp to encrypt a message for Daisy. Goofy does not want to repeat the authentication step every time he sends a message to Daisy, so he opens his notebook and writes down ("Daisy", pk) to confirm that "Daisy" owns the public key pk.

Every next time Goofy will want to send a message to Daisy, he will fetch pk, pp from her website again. He will then check whether his notebook has an entry ("Daisy", pk). If such entry exists, Goofy will know that Daisy still uses the same public key. If such entry does not exist, then Daisy must have changed her public key recently, so Goofy will treat it as the first time he messages Daisy: he will authenticate the new contact data with the CA prior to using it to send a message.

(a) Assume the Evil Queen can manipulate the information that Goofy fetches from Daisy's website (e.g., by spoofing the website, or by intercepting and mauling traffic between Goofy and Daisy's website). Show how the Evil Queen can break the privacy of the PKE-encrypted traffic sent from Goofy to Daisy, such that *any eavesdropper* can trivially read all messages that Goofy encrypts for Daisy. *It is <u>not</u> required that Daisy can still decrypt Goofy's messages using algorithm* Dec.

(b) Improve the idea from (a) so that *only the Evil Queen* can trivially read all messages that Goofy encrypts for Daisy. Any other parties that can inspect the PKE-encrypted traffic sent by Goofy should not be able to learn *any* information about the messages he encrypts for Daisy. *It is <u>not</u> required that Daisy can still decrypt Goofy's messages using algorithm* Dec.

In January 2020, Microsoft fixed the *CurveBall* (CVE-2020-0601) vulnerability [1] in Windows CryptoAPI that allowed to use the above attack idea in order to forge ECC-based signatures on malicious executables. (For the sake of this exercise, we adapted the idea to attack the *privacy of encryption* instead of the authenticity of signatures).

Problem 3 (Key separation). To save space and key material, it might be tempting to re-use the same keys for several purposes.

(a) Explain why, generally, this is a very bad idea.

(b) Consider a scenario where Alice uses the key-pair $sk = d$, $pk = (e, N)$ to encrypt her messages using RSA PKCS#1 v1.5 (cf. Lectures 22-26). Suppose the decryption implementation

provides a basic Bleichenbacher oracle. That is, an adversary is equipped with an oracle that, on input c, computes $m = c^d \bmod N$ and returns 1 if the most significant two bytes of m are "0x00 0x02", and returns 0 otherwise.

Now also suppose that, to avoid the cost of paying for two certificates, Alice decides to re-use the same key-pair to digitally sign messages using the RSA-PSS scheme (cf. Lectures 27-28).

Show how an adversary can forge Alice's RSA-PSS signature on an arbitrary message m using the oracle. Comment on the complexity of your attack in terms of the expected number of oracle calls.

Hint: Recall that if c is a valid ciphertext w.r.t. RSA PKCS#1 v1.5 encoding, then the adversary can efficiently recover $c^d \bmod N$ by running Bleichenbacher's attack. How can the adversary use this facility to forge a signature?

Acknowledgements. This exercise sheet is in part inspired by (and adapted from) problems by Simon Blackburn at Royal Holloway, University of London, as well as the book "A Graduate Course in Applied Cryptography" by Dan Boneh and Victor Shoup.

References

[1] Curveball (cve-2020-0601). https://blog.trendmicro.com/trendlabs-security-intelligence/an-in-depth-technical-analysis-of-curveball-cve-2020-0601/.

[2] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography*. Online, version 0.6 edition, Jan. 2023.