Content of these lecture notes. See Pihla's lecture notes for the definition of public-key encryption (PKE) and a discussion of black-box proofs. IN these lecture notes, we briefly discuss IND-CPA security (Section 1) for public-key encryption schemes and then present four sections with supplementary material.

Section 2 discusses why it's problemantic if a PKE is deterministic—it cannot be IND-CPA-secure then.

Section 3 describes why RSA-OAEP was introduced, a concrete (real-world) public-key encryption scheme.

Section 4 discusses the random oracle model.

Section 5 contains reading material for non-black-box constructions/proofs (cf. Pihla's discussion).

Section 6 contains pointers to studies of black-box constructions/proofs (cf. Pihla's discussion).

Section 7 briefly discusses post-quantum cryptography.

1 IND-CPA security for PKE

Public-key encryption is the public-key analogue of symmetric-key encryption (Lecture 5). The main difference is that now, there is not only a single symmetric key k for encryption and decryption, but instead, there are two keys, a public-key pk for encryption and a symmetric key sk for decryption.

Algorithms. A pke consists of three PPT algorithms $pke.\mathsf{kgen}$, $pke.\mathsf{enc}$ and $pke.\mathsf{dec}$ where

Key Generation $(pk, sk) \leftarrow\$ pke.\mathsf{kgen}(1^\lambda)$ takes the security parameter 1^λ and outputs a public key pk and a related secret-key sk.

Encryption $c \leftarrow\$ pke.\mathsf{enc}(pk, x)$ takes the public-key pk, the message x and returns a ciphertext c.

Decryption $x \leftarrow\$ pke.\mathsf{dec}(sk, c)$ takes the secret-key pk, a ciphertext x and returns a message m.

Correctness requires that decryption with the right sk allows to recover a message x from the ciphertext which was generated using pk, i.e., $\forall (pk, sk) \leftarrow\$ pke.\mathsf{kgen}(1^\lambda)$, $\forall x \in \{0,1\}^* : \forall c \leftarrow\$ pke.\mathsf{enc}(pk, x)$,

$$pke.\mathsf{dec}(sk, c) = x.$$

Domains. Note that message x and ciphertext c are sometimes from other domains than bitstrings. In fact, since public-key encryption is mostly algebraic, message x and ciphertext c often come from some set that is related to some algebraic structure. For example, for RSA, x and c both from from $\{1, .., N-1\}$, where N is a natural number that is part of the public-key.

Security. The magic property of secure public-key encryption is that the *confidentiality* of the encrypted messages is guaranteed even though given the public-key, *everyone* can generate ciphertexts on their own.
The definition of confidentiality is analogous to how we defined it in the symmetric-key case, namely via indistinguishability against chosen-plaintext attacks (IND-CPA) where real encryptions of messages m_i should be indistinguishable from encryptions of zero strings $0^{|m_0|}$. Now, one difference to the symmetric-key case is that the attacker can actually also make use of $pke.\mathsf{enc}$ using the same public-key pk than the game, because the attacker can obtain the public-key via the GETPK oracle. We now define IND-CPA security for public-key encryption formally.

Definition 1.1 (IND-CPA)**.** A public-key encryption scheme *pke* is IND-CPA-secure if for all PPT adversaries $\mathcal{A}$

$$|\Pr[1 = \mathcal{A} \to \texttt{Gind-cpa}^0_{pke}] - \Pr[1 = \mathcal{A} \to \texttt{Gind-cpa}^1_{pke}]| \text{ is negligible in } \lambda.$$

```
$\texttt{Gind-cpa}^0_{pke}$

$\mathsf{GETPK}()$
if $pk = \bot$ :
    $(pk, sk) \leftarrow\$ pke.\mathsf{kgen}(1^\lambda)$
return $pk$

$\mathsf{ENC}(x)$
if $pk = \bot$ :
    $(pk, sk) \leftarrow\$ pke.\mathsf{kgen}(1^\lambda)$

$c \leftarrow\$ pke.\mathsf{enc}(pk, x)$
return $c$
```

```
$\texttt{Gind-cpa}^1_{pke}$

$\mathsf{GETPK}()$
if $pk = \bot$ :
    $(pk, sk) \leftarrow\$ pke.\mathsf{kgen}(1^\lambda)$
return $pk$

$\mathsf{ENC}(x)$
if $pk = \bot$ :
    $(pk, sk) \leftarrow\$ pke.\mathsf{kgen}(1^\lambda)$
$x' \leftarrow 0^{|x|}$
$c \leftarrow\$ pke.\mathsf{enc}(pk, x')$
return $c$
```

2 Deterministic PKE

If a public-key encryption (PKE) is deterministic, then the situation is even worse than in the symmetric-key encryption case. In the symmetric-key encryption case, the leakage of deterministic encryption means that an observer can spot repetitions. However, in deterministic *public-key* encryption, an attacker can always check their guess for the content of a message against the actual message. That's a much stronger capability, Therefore, randomness is even more important than in the symmetric-key encryption case.

Textbook RSA Textbook RSA usually describes a deterministic function, also knows as a *trapdoor function*. Namely, we have a key generation algorithm $\mathsf{tf.kgen}$ which outputs a pair (pk, sk) and then $f(pk, x)$ is a one-way function, i.e., for all PPT adversaries $\mathcal{A}$, the probability that the following experiment returns 1 should be negligible:

```
$\mathsf{Exp}_{\mathsf{TDF}}(1^\lambda)$
$(pk, sk) \leftarrow\$ \mathsf{kgen}(1^\lambda)$
$x \leftarrow\$ \{0,1\}^\lambda$
$y \leftarrow f(pk, x)$
$x' \leftarrow\$ \mathcal{A}(pk, x, 1^\lambda)$
if $f(pk, x') = y$ :
  return 1
return 0
```

In RSA, the public-key *pk* consists of a modulus N and an exponent e, and the secret-key *sk* consists also of the modulus N as well as another exponent d. In practice, for e, the value 65537 is popular due to its binary representation since it contains only two ones, so that calculating exponentiation is quite easy. Note that $e = 3$ can be a bad choice due to the Coppersmith attack `https://en.wikipedia.org/wiki/Coppersmith%27s_attack#H%C3%A5stad%27s_broadcast_attack`

Trapdoor function In cryptography, the RSA function is referred to as a *trapdoor function* or a *trapdoor permutation* (since it is bijective). Outside of cryptography, the RSA function is also referred to as "encryption function". This can be quite misleading terminology, as RSA on its own is not a full encryption scheme whereas the name seems to suggest that RSA alone is good encryption...

3 Encrypting with RSA

Encrypting with RSA is not an easy task. Firstly, one needs to add randomness in some way (to be discussed in more detail shortly), and secondly, we need to cope with the issue that one-wayness is an extremely weak assumption, as we have seen in the first lecture of the course. Therefore, proving IND-CPA security merely based on a one-wayness definition seems already quite challenging conceptually. One could go through hardcore bits `http://www.wisdom.weizmann.ac.il/~oded/X/acgs.pdf`, but this seems to yield a quite inefficient encryption system.
We now first describe RSA-OAEP and then discuss the different approaches towards proving RSA-OAEP. Before turning to RSA-OAEP, just a quick discussion about a previous approach to randomizing RSA which failed quite strongly.

Padding Oracle attacks on RSAES-PKCS1-v1_5 The Bleichenbacher attack on RSAES-PKCS1-v1_5 `https://en.wikipedia.org/wiki/PKCS_1` is a variant of the padding oracle attack which we have seen in the *before the lecture* part of lecture 4. Namely, the nice algebraic properties of RSA lend themselves as the basis for attacks, if the padding is not carefully done. For RSAES-PKCS1-v1_5, it was shown that a partial decryption oracle (i.e., an oracle which tells whether the padding is correct or not) can be used to recover a message encrypted using RSAES-PKCS1-v1_5 bit by bit.

RSA-OAEP See `https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding` for the description of the RSA-OAEP padding scheme.

4 Random Oracle Proofs

Bellare and Rogaway who proposed the OAEP `https://cseweb.ucsd.edu/~mihir/papers/oaep.html` originally carried out their proof in the *random oracle model*, i.e., they assumed that the functions H and G are implemented by oracles which behave as public, truly random functions, so-called *random oracles* (RO). While it is clear that a (deterministic, public) hash-function cannot behave like a true random function and cannot be indistinguishable from a truly random function (since, unlike a PRF, a hash-function does not have a secret key), modeling hash-functions as random oracles is popular in the research community, since it simplifies proofs. Namely, instead of a (very weak) one-wayness assumption (as the one we previously described for trapdoor functions), in the proof, one can use the much stronger assumption that $\mathsf{RO}(x)$ is uniformly random and secret as long as the attacker does not know x. Bellare and Rogaway deem random oracles a solid methodology for practical analysis `https://cseweb.ucsd.edu/~mihir/papers/ro.pdf`, whereas other researchers collect the flaws and problems of the methodology. Canetti, Goldreich and Halevi (CGH) were the first to demonstrate technical and conceptual flaws in the random oracle model methodology `https://eprint.iacr.org/1998/011.pdf`. The article essentially shows that there are cryptographic schemes which can be proven secure in the random oracle model, but are insecure whenever we replace the random oracle by a concrete function. I.e., CGH show that *there is no concrete hash-function that makes the scheme secure.* Thereby, CGH establish that the random oracle methodology might make us believe that certain schemes are secure, while they are blatantly insecure. The CGH paper is a good example of how controversial the random oracle methodology is discussed in the field: Each author wrote their own conclusion, since each of the three interpreted the facts brought forward in the paper differently. CGH was then later criticized for providing only contrived counterexamples, whereas, so the critics' opinion, "natural" cryptographic schemes would not show such strange behaviour as the ones presented in CGH. For more recent works on understanding random oracles, see, e.g., `https://eprint.iacr.org/2013/424` and `https://eprint.iacr.org/2014/867`.

(Un)instantiability proofs for RSA-OAEP It was sought to establish that security of RSA-OAEP can *only* be proven in the random oracle, perhaps pointing out another weakness of the random oracle model. See, e.g., `https://link.springer.com/chapter/10.1007/978-3-642-01001-9_23` for an uninstantiability result (i.e., a result showing that the random oracle might not be instantiable by any concrete hash-function) by Kiltz and Pietrzak. However, a little later, Kiltz, O'Neill, and Smith showed `https://link.springer.com/content/pdf/10.1007/978-3-642-14623-7_16.pdf` that the hash-functions in RSA-OAEP can be securely instantiated when making stronger assumptions on the RSA function than merely assuming that it is a one-way function.

5 Non-black-box constructions and reductions

Example 1: Zero-Knowledge In 2008, in a celebrated breakthrough result, Barak showed that treating the adversary in a *non-black-box* way might sometimes allow us to prove statements which were impossible to prove in a black-box way [?]. His idea emerged in the context of certain zero-knowledge proofs and the difficulty of showing security against a malicious verifier (who wants to break the zero-knowledge property). On the very high-level, the idea, here (described on page 23 of the article), is that if knowing the code of the verifier allows to compute a *commitment* to the code of the verifier, which later allows to show that the verifier, indeed, behaved correctly. The details of the proof are beyond the scope of this course—but if you are curious about this paper, then this article would be a fantastic choice for a presentation in the context of the course MS-E1687 Advanced Topics in Cryptography `https://mycourses.aalto.fi/course/view.php?id=32919`.

Example 2: Obfuscation Barak's technique did not lead to a broad use of non-black-box use of an adversary. However, non-black-box use of a cryptographic *primitive* became a very popular technique after 2013 in cryptography. Namely, in 2013, a breakthrough candidate construction by Garg, Gentry, Halevi, Raykova, Sahai and Waters (GGHRSW `https://eprint.iacr.org/2013/451`) convinced the cryptographic community that *obfuscation* of arbitrary circuits might be possible, after all. This was a big surprise due to an impossibility result from 2001 by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang (BGIRSVY `https://www.wisdom.weizmann.ac.il/~oded/p_obfuscate.html`) which showed that so-called *virtual black-box* obfuscation for general circuits is impossible. BGIRSVY also propose a very weak notion of obfuscation known as *indistinguishability obfuscation* and believed to be rather useless, see, e.g., `https://www.wisdom.weizmann.ac.il/~oded/MC/136.html`. However, Sahai and Waters `https://eprint.iacr.org/2013/454` showed that indistinguishability obfuscation is actually a very useful primitive, and with the GGHRSW candidate construction together, the cryptographic community started on fruitful and lively research on (a) applications of indistinguishability obfuscation and (b) constructions based on reasoneable assumptions. Only this year, indistinguishability obfuscation has been based on quite well-founded assumptions `https://eprint.iacr.org/2021/1334`. You can ask Kirthi, Estuardo or Chris if you are interested in more background on the topic of obfuscation.
But what is the connection between obfuscation and non-black-box techniques? Obfuscation takes a circuit/program and turns it into an non-intellegible version of that circuit/program—but it maintains the input-output behaviour. In order to use obfuscation, we need to describe our cryptographic primitive concretely as a circuit/program. Therefore, when relying on obfuscation, the underlying cryptographic primitives cannot be treated in a black-box way.

6 Black-box proofs

The first authors to study the power of *black-box proofs* in cryptography were Impagliazzo and Rudich who showed that one cannot build public-key encryption from one-way permutations in black-box way `https://link.springer.com/content/pdf/10.1007`. Later, Reingold, Trevisan and Vadhan sought to cleanly characterize which techiques are captured by black-box impossibility results à la Impagliazzo-Rudich and thus defined a classification of notions of reducibility between cryptographic primitives `https://omereingold.files.wordpress.com/2014/10/blackbox.pdf`. This work was later refined by Baecher, Brzuska and Fischlin `https://eprint.iacr.org/2013/101`. Nowadays, I also think that a good definition of *black-box* is to say that everything which can be proven via this *package methodology* (which Osama briefly presented) is a black-box proof, see the Crypto Companion or `https://eprint.iacr.org/2018/306`.

7 Post-Quantum Cryptography

The National Institute of Standards and Technology (NIST) started a competition for post-quantum secure public-key encryption schemes. In fact, the actual competition concerns post-quantum secure *key encapsulation mechanisms* (KEM). These are public-key encryption schemes which only encrypt *random* messages (i.e., *keys*) instead of chosen messages. Once two parties share a secret random message, they can use this random message as a key for a symmetric encryption scheme—this is known as the KEM-DEM approach, see, e.g., Section 4 in `https://eprint.iacr.org/2018/306`. I did not say much about the NIST competition in the lecture, but if you are interested in the current state-of-the-art of post-quantum secure public-key encryption, then this might be a good starting point: https://www.nist.gov/news-events/news/2020/07/nists-postquantum-cryptography-program-enters-selection-round