General notes on cryptographic hash functions
Important properties of a cryptographic hash function h:
- First preimage resistence: Given fixed h(m), it is computationally difficult to find m.
- Second preimage resistance: Given fixed m, it is computationally difficult to find m'≠m such that h(m)=h(m')
- Collision resistance: It is computationally difficult to find any m and m', with m≠m' such that h(m)=h(m')

Here is an example of a ***broken*** hash function that one student mentioned:
https://en.wikipedia.org/wiki/MD5
Some further goodies about MD5:
http://web.archive.org/web/20071226014140/http://www.cits.rub.de/MD5Collisions/
https://www.mathstat.dal.ca/~selinger/md5collision/

Another example is SHA-1, which is still widely used in many applications, but fortunately being used less and less.
See https://shattered.io/ for more details

Breakout Session 1 on RSA and Hash Functions

When broadcasting a message to multiple people, you may be able to use the Chinese Remainder Theorem to reveal the message.

Question:
Q: What padding schemes are used in the real world?
A: See:

- https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding
- https://en.wikipedia.org/wiki/PKCS

Q: During the videolecture Tanja explained how to compute RSA signatures on a message m like this: Compute h(m)^d%n, where h is some k-bit hash function. We require 2^k<n (using h(m)=h0,h1,...,h_{k-1} as sum_{i=0}^{k-1}h_i*2^i).
Can you explain in your words what she means by this?

A: The output of a hash function is a bit string of length k. We can view this bit string as an integer encoded in binary (little-endian in this case, for the CS students), and interpret it as a base 10 integer (modulo n) using the summation given in the question. We then use this number to compute the signature.

Breakout Session 2 on RSA and Hash Functions

Q:When signing, why do we use the hash functions. how is signing different to encrypting and decrypting a message with 'normal RSA'?
A: Cryptographic hash functions output "random"-looking outputs that eliminate algebraic relationships between messages that could be used in attacks.

Q: Can you explain general properties of hash functions?
A: Hash functions in the most general sense map arbitrarily long sequence of bits to a fixed-length sequence of bits. For cryptographic hash functions we require the three properties mentioned above. I gave some motivation for these properties using RSA signatures.

Breakout Session 3 on RSA and Hash Functions

Q:How are the dangers of the version of RSA Tanja has shown mitigated in real life?

A: There are many mitigations used in real systems. Just a few of them: we use hashing in signatures to eliminate algebraic relationships between messages. We use padding to make sure that exponentiated values are large enough to be reduced mod n. We also use padding to introduce randomness so that encryption is not deterministic.

Q: Are there any alternatives to RSA if you want to use public key cryptography?
A: YES. Too many to mention here. You'll learn about some of them in this course.

Q: Why do we only use hashes for RSA signatures?
A: There are two sides to this coin as I think about it: A desired property of cryptographic hashes is that they look like "random" functions, and so they eliminate any structure between related messages that is preserved during the exponentiation mod n. This adds security against some kinds of attacks for signatures. However, they are also are one-way functions (i.e. they have first pre-image resistance), and so if i encrypt a message m by computing h(m) ^ e % n, then even if the owner of the private key decrypts this ciphertext, they cannot recover m from h(m). So hashing wouldn't be very useful in this context.

Q: During an earlier lecture Tanja proved why the RSA system is sound, i.e. m = m'. I understand the case where gcd(m,n) = 1, but I am a little bit confused by the case where m = r*p and thus gcd(m,n) is not 1. Can you maybe explain that part of the proof?

A: Rough sketch: if m not in Z*_n, we still want that m^{ed} = m % n . Remember ed = 1 + k * phi(n), so m^{ed} = m^(1 + k * phi(n)) = m * m ^ k * phi(n). Given that phi(n) = (p-1) * (q-1), we can look at m^k*phi(n) mod p and mod q. You should be able to use this and CRT to show that this is 1 mod n. So you get that m ^ {ed} = m mod n, even if m not in Z*_n.