Exercise 1

1. We covered (cryptographic) hash functions in the lecture. What are two general properties of hash functions? What properties constitute cryptographic hash functions? Explain them in simple words.

Solution:

- General Properties:
  - Compression: Arbitrary input length, fixed output length
  - Ease of computation: Given h and x, it is easy to compute $h(x)$
- Cryptographic Properties:
  - Pre-image resistance: Impossible to find the original x if only $h(x)$ is given
  - Second pre-image resistance: Impossible to find x' if x and $h(x)$ are given $(h(x') == h(x))$ (and $x' \neq x$)
  - Collision Resistance: Impossible to find x and y s.t. $h(x) == h(y) \wedge x \neq y$
  - Hiding: Impossible to determine x in $h(r||x)$ given only the result of the hash. r is a random number. (Hiding is not a required property for being a cryptographic hash function. We state it here out of completeness.)

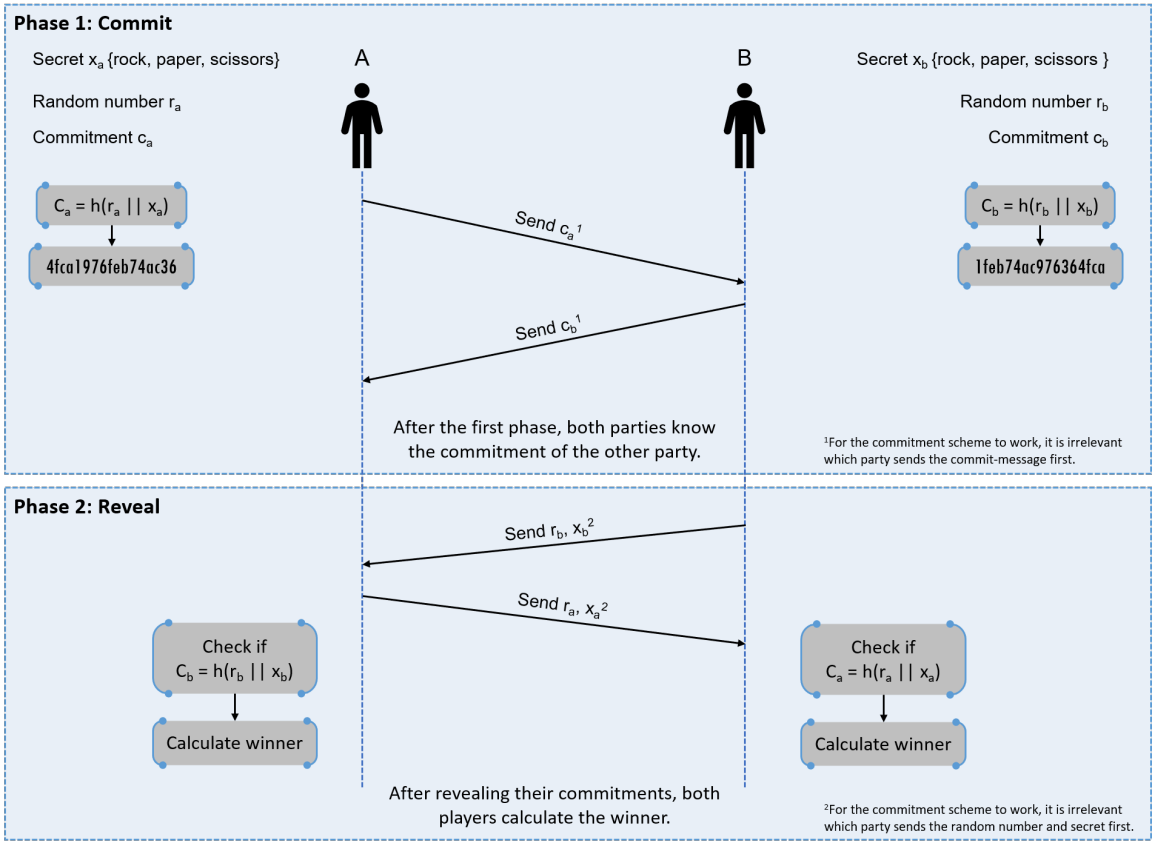
2. Alice wants to play Rock, Paper, Scissors with Bob via an internet connection. Both want to win and are afraid their opponent will just wait for the message and respond accordingly, as no one knows when the other sent the message. Create a scheme in which it is not possible to cheat.

Solution:

We use the so-called commit and reveal-scheme. It consists out of two phases:

- The commitment-phase: Both Alice and Bob commit themselves to either rock, paper, or scissors by hashing it with a random number. $commitment = (h(rock||randomnumber))$ This commitment is then sent to the other party. Because the information (in the above example rock) is hashed, it is computationally infeasible to guess whether the other party has chosen rock, paper, or scissors.
- The reveal-phase: After both parties received the message from the other party, they reveal their choise and the random number. To reveal the information, they just send it to the other person. Each Alice and Bob verify if the received information matches the hash by calculating the commitment itself. As of the properties of cryptographic hash functions, both parties are not able to change their initially commited rock, paper or scissors.

In the following image, we display a graphical example of the commit and reveal-scheme.



3. It is best practice to store only hashed user login information. Often, the hash not only contains the password itself, but also some additional user-specific information. Explain why this is a better approach than just storing user-passwords.

Solution:

Be careful: The correct storing of passwords is a complex process which we do not cover in detail. In this exercise, we just want to give you an impression of other usages for hash-functions.

If a database breach occurs, attackers can access all data, thus also stored passwords. To make it impossible for attackers to obtain the actual passwords of the users, the hashing of the passwords should be done properly. Additional user-specific information (a unique random string is recommended) is called salt. It prevents the creation of lookup-tables. Lookup-tables are large databases which contains pontential passwords and the corresponding hashes. An attacker could just take the hashes from the obtained database and search for hash-matches (resulting in a found password).

Further remarks: It is best-practice to use a user-specific random string and not rely on user submitted information. Please refer to the OWASP password storage cheat sheet for further information and best practices.

4. We want to create a search puzzle. Use the puzzleID "BBSE_E01" and the SHA_256 hash function. Assume d to be "0000f00".

   (a) What is the value x that solves the puzzle? How long does your computer execute until it finds a result? Select your favorite programming language and develop this search puzzle.

Solution:

Code:

```php
<?php
$puzzleID = "BBSE_E01";
$d = '0000';
$x = 0;

while(true)
{
    $result = hash("sha256",$puzzleID.$x);
    if(substr($result, 0, strlen($d)) === $d) { $res = $x; break;}
    $x = $x+1;

}
echo "Found: x:".$res." returns hash value ".$result;

?>
```

$7852 \rightarrow sha_{256}(\text{BBSE\_E017852}) = 000072d930284346c3d54c5cda0306900078c9be695b24d899d5ebeb2057ebcf$

Further information about the program: The used language is PHP. In this case, the concatenation in the first line of the while-loop works in PHP with a dot (.) instead of our known notation (||). In this simplified program, we do not compare the numbers (difficulty and hash), but instead we check if the hash starts with a certain amount of zeros. If that is the case, the while loop exits and displays the found number.

   (b) Three computers (hashing power $A = 50\%, B = 30\%, C = 20\%$, overall 100 000 hashes per second) participate in this search puzzle. All computers iterate through all x with $x + 1$. Which one wins the search puzzle?

Solution:

The computer A wins, as he will reach x=7852 as the first computer. It is the fastest.

   (c) Is there a way for the losing computers to increase their chances of winning?

Solution:

They have to manipulate the search puzzle. Possible ways are:

- Do not iterate with x+1, but start with another number for x
- Do assign x a random value every time

```php
<?php
error_reporting(0);
$puzzleID = "BBSE_E01";
$d = '0000';
$x = 0;

while(true)
{
    $result = hash("sha256",$puzzleID.$x);
    if(substr($result, 0, strlen($d)) === $d) { $res = $x; break;}
    $x = rand();

}
echo "Found: x:".$res." returns hash value ".$result;

?>
```

(d) Can we design the puzzle in such way that the users win in accordance to their hashing power?

Solution:
Yes. We introduce a variable in the $PuzzleID$ that is user-dependent. For example, the hashing process could look like this: $Hash(PuzzleID||computerName||x)$

5. Take a look at the Merkle tree in Figure 1. It contains four data elements. How would you create a Merkle tree with five elements building upon the given structure?

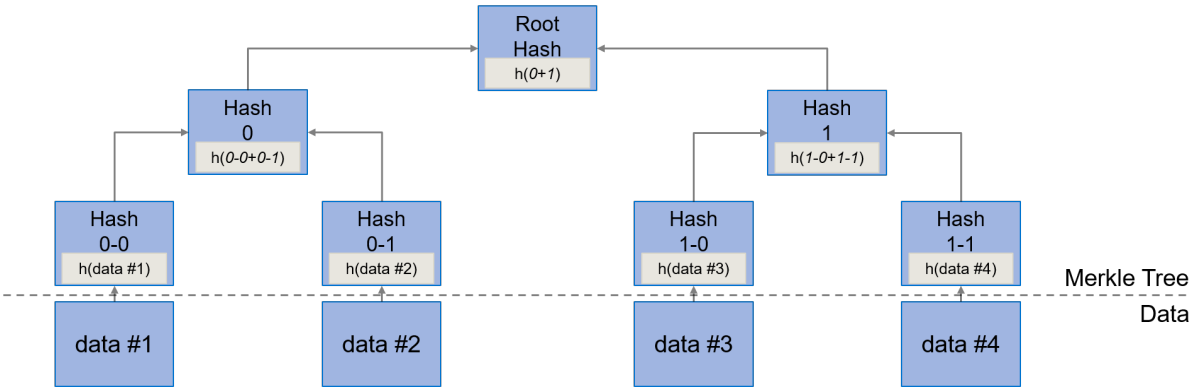


Figure 1: Merkle Tree

Solution:
Tree:

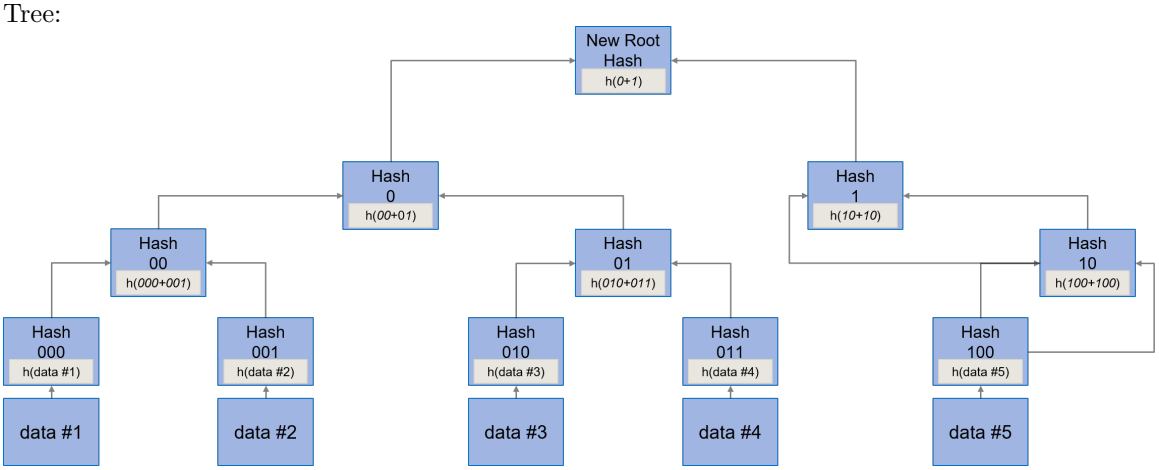


Figure 2: Merkle Tree - Solution

If elements are missing, they are just replaced with the existing ones. In our case, element #6 is missing to calculate hash 100. Therefore, element #6 is replaced with element #5 (as it is available). Afterwards, the hash 11 is missing to calculate hash 1, therefore it is replaced with element 10. This hash 0 is concatenated with hash 1 and a new merkle root tree is calculated.
Please note, that systems do not "recalculate" Merkle trees by adding or removing elements afterwards. They are calculated once, otherwise they are discarded. Once a Merkle tree is included in a block which is included in a Blockchain, the tree cannot be changed anymore.

6. The Technical University of Munich (TUM) offers a service for companies and other interested parties to validate certificates it has issued. Therefore, the administration decided to:

   1. Hand out a PDF-document to each student with his diploma and
   2. Publish all hashes of these diplomas on www.certificates.tum.de. This website provides an easy way for anyone to validate the document.

   If a student wants to prove she got a certificate of TUM, she gives the PDF-document to the company. The company hashes the file and verifies that this hash is indeed published at www.certifcates.tum.de.

   (a) The administration decided that it might not be the best idea to just publish all hashes, as it reveals the number of diplomas issued in a certain time period. Design a process using hashing algorithms such that:

      1. The integrity of the diplomas is still ensured
      2. It is unknown to a third party how much diplomas are issued

      Explain what additional information is required to properly validate a certificate.

      Solution:
      Please note that there are many valid solutions for this exercise. In session we discussed other possibilities, such that all hashes are stored in a private database and a server-side application checks if the entered hash is in fact stored in the database. This exercise is merely a constructed example to give you an idea how the concepts introduced in the "Cryptographic Basics"-lecture could be applied. The intention of this exercise is of course to later rely on a decentralized network like a Blockchain.
      One possible way is to create Merkle trees. The leaves of the tree are the hashes of the issued diplomas. The root hash ensures the integrity of the diplomas. There are two downsides with this approach:

      - Additional information has to be given to the students as they need to be able to prove that their hash is in the root hash. This is done via the inclusion proof for Merkle trees (see lecture slide 18).
      - A new (additional) tree has to be created to add new diplomas. The existing trees cannot be modified, otherwise we would have to resend the additional information (see point above) to the students each time we update the Merkle Tree.

   (b) The administration wants to extend the tool such that it is possible to invalidate hashes. It discusses if it should just publish the revoked hashes, however, this would reveal the number of certificates it has revoked (which can be sometimes embarrassing). First, explain why it is not possible to remove the hash out of the Merkle Tree. Then design a scheme in which it is possible to validate if a certain hash was revoked without revealing additional information about the number of revoked certificates.

      Solution:

It is not possible to remove the hash out of the Merkle tree, as the required information for verification would change and it would be required to resend the additional information (inclusion proof, see question above, downside 1) to the students.
Instead of revoking one certificate, the university could revoke a certain number of hashes (which do not belong to any certificate) every week. With this, it could hide the fact that it has revoked a certificate because it is unclear to external parties if a real diploma was revoked with these hashes or not.

7. Why are public keys of identities hashed before they are used as an address? Name two reasons and explain.

Solution:

- Public keys have a high character count. Hashing allows to reduce the storage requirements.
- If quantum computers are feasible, popular signature schemes (like RSA or ECC) will likely be broken. In order to prevent the generation of private keys out of public keys, the public keys are hashed. The hash functions are not affected of quantum computers. A user controlling funds can store them on a public key hash, as the generation of the public key out of the hash is computationally infeasible. The user has to consume the complete transaction, sending the change to a new address which was not revealed before. Therefore, address reuse is not possible / not advised.