WEB3CLUBS FOUNDATION LIMITED

Course Instructor: DR. Cyprian Omukhwaya Sakwa
PHONE: +254723584205 Email: cypriansakwa@gmail.com

Foundational Mathematics for Web3 Builders

Lecture 26

June 18, 2024

Example 3

$p \times q = n = \underline{7081}$ $5629 \equiv x^2$ ✓

Eve attempts to impersonate Cyprian in the example 2. Alex challenges Eve to prove she knows x.

- Eve knows $n = 7081$ and $5629 \equiv x^2 \bmod n$, since these were published by Cyprian, but she does not know x.
- Eve chooses $u_1 = 170$ and then computes $(u_1)^2 = 170^2 \bmod 7081 = 576$.
- She then computes $(u_1^2)^{-1} = 576^{-1} \bmod 7081 = 2053$. Then computes

  $$u_2^2 = x^2 \cdot (u_1^2)^{-1} = 5629 \cdot 2053 \bmod 7081 = 145.$$
- Eve sends to Alex $x_1 = (u_1)^2 = 576$ and $x_2 = (u_2)^2 = 145$.
- Alex checks that $x_1 \cdot x_2 \bmod 7081 = 576 \cdot 145 \bmod 7081 = 5629 = x^2$.
- Alex sends Eve the message "send me $\sqrt{x_1}$."

Example (conti...) 15 81

- Eve responds with the message $\sqrt{x_1} = 170$. Eve passes this round.
- Alex sends Eve another message "send me $\sqrt{x_2}$."
- Eve is unable to find $\sqrt{x_2}$.
- Thus, Eve fails the challenge, and now Alex knows she is an impostor.

1.1 To confirm that this protocol is a zero knowledge proof, we examine three things:

a) The test must be complete: Cyprian can always pass it.

b) The test must be sound. That is, to pass the test, intruder Eve must know the value of x. To pass the test, Eve must be able to compute $\sqrt{x_1}$ and $\sqrt{x_2}$ modulo n. If she can do this, then she can compute x, since it's the same as knowing it. If Eve does not know x, she can only provide one of the two values $\sqrt{x_1}$ or $\sqrt{x_2}$ when challenged by Alex, giving her a 50% chance of passing the test.

c) The test needs to be zero-knowledge, meaning intruder Eve can't learn the secrets by eavesdropping in on actual conversations between Cyprian and Alex.

Algorithm 2 (Zero knowledge identification scheme)

Let $n = pq$ represent the product of two huge prime numbers. Let Cyprian have the secret numbers $s_1, s_2, s_3, \cdots, s_k$ and $v_i \equiv (s_i)^{-2} \bmod n$, with $\gcd(s_i, n) = 1$. Alex receives the numbers v_i and checks to see if Cyprian recognizes the numbers $s_1, s_2, s_3, \cdots, s_k$. Both Alex and Cyprian proceeded as follows:

1. First, Cyprian selects a random number r, computes $x = r^2 \bmod n$ and sends x to Alex.
2. Alex then selects numbers $\{b_1, b_2, \cdots, b_k\} \in \{0, 1\}$ and sends them to Cyprian.
3. Cyprian calculates $y \equiv r(s_1)^{b_1}(s_2)^{b_2}(s_3)^{b_3} \cdots (s_k)^{b_k} \bmod n$.
4. Alex checks that $x \equiv y^2(v_1)^{b_1}(v_2)^{b_2}(v_3)^{b_3} \cdots (v_k)^{b_k} \bmod n$.
5. Steps (1) through (4) should be repeated multiple times (about 20–30 times), using a different r each time.

Note

a) It is worth noting that the robustness of this identity-verification method is contingent upon the difficulty of computing square roots $\underline{\text{modulo } n}$.

b) It's also crucial to remember that, contrary to how the phrase is sometimes used, the aforementioned system is not a "proof of identity." Following the zero-knowledge proof protocol, the only information that can be discovered by anybody is this: Alex is now confident that the person he is speaking with is aware of the secret number x.

Example 4 ✓

In this identification technique, we suppose that there is a smart card owned by, say, Cyprian, a card reader machine owned by, say, a bank, and a third party, known as the third trust party (TTP).

a) The TTP selects $n = pq$, where p and q are two huge primes and $p \equiv q \equiv 3 \bmod 4$. It then computes the PIN number for Cyprian's smart card, which is
$$PIN \equiv x^2 \bmod n.$$

b) The TTP computes the square root x of ID using the prime factorization of n and stores it on the smart card's secure memory segment. The TTP should also make n public, while keeping p and q private. The smart card now holds the information (PIN, n, x), whereas the card reader has the information n.

Example (conti...)

c) The Smart Card or the card holder Cyprian enters the PIN number into the card reader:

$$\text{Card/Cyprian} \xrightarrow{\text{PIN}} \text{Card reader}$$

d) Card/Cyprian creates a random r, calculates $t \equiv r^2 \bmod n$, and transmits it to the card reader:

$$\text{Card/Cyprian} \overset{t}{\rightarrow} \text{Card reader}$$

e) The Card Reader randomly selects $e \in \{0, 1\}$ and sends it to Cyprian:

$$\text{Card/Cyprian} \overset{e}{\rightarrow} \text{Card reader}$$

f) Card/Cyprian calculates

$$u \equiv r \cdot x^e \bmod n$$

and sends it to the card reader.

$$\text{Card/Cyprian} \overset{u}{\rightarrow} \text{Card reader}$$

Example (conti...)

g) The Card Reader determines whether or

$$u^2 \equiv t \cdot \mathsf{PIN}^e \bmod n$$

h) Steps (4) through (7) should be repeated for different r values. If each time

$$u^2 \equiv t \cdot \mathsf{PIN}^e \bmod n$$

then the TTP issues the card. That is, the Card Reader is convinced that the Card has stored x, the square root of the PIN modulo n.

Example 5 (How two users could use a ZKP to transact.)

- Person A (prover) wishes to perform a transaction using the privacy coin ZCash so that no blockchain viewers may extract any transaction details (e.g., the amount transferred, the address to which the transfer is made, and so on).
- Person B (verifier) expects to receive ZCash from Person A, but does not know their private information, such as the total amount of assets in Person A's wallet.
- Person A encrypts and sends the transaction to the blockchain.
- Person A sends a ZKP together with the encrypted transaction to confirm that it is legitimate.
- Nodes on the blockchain verify the ZKP to confirm that Person A's encrypted transaction is legitimate.
- If the ZKP is valid, Person B will agree.
- The ZCash is transferred from Person A's account to Person B's account.

1.2 Classification of Proof systems

- Proof systems are classified according to the quantity of messages that can be transmitted.
- ZKPs exist in two types: interactive and non-interactive.
- Interactive ZKPs require numerous rounds of back-and-forth communication between the prover and verifier.
- Non-interactive ZKPs require only one round of communication – the prover sends a single message to the verifier, and they do not need to be online at the same time for this to happen.
- A proof system is considered succinct if the size of the proof is less than the witness required to generate the proof. A witness is the secret information that needs to be proven and authenticated. Computationally sound proof systems require the prover's computational capabilities to have polynomial bound.

- Interactive proofs are uncommon in blockchain-based systems since they are wasteful and need two parties to be online simultaneously. As a result, ZK-SNARK and ZK-STARK, are the two most potent cryptographic methods available for ZKPs utilized in non-interactive configurations
- zk-SNARKs and zk-STARKs are based on sophisticated mathematical ideas such as polynomial equations, elliptic curves, bilinear pairings, and cryptographic protocols.

1.3 zk-SNARK

- The acronym zk-SNARK, which stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, refers to a proof construction in which one can prove possession of specific information, such as a secret key, without revealing that information or requiring any interaction between the prover and verifier.

- Zk-SNARKs serve as proof that one user possesses knowledge without disclosing it.
- This is a complex yet computationally efficient mathematical procedure employed in blockchains to protect user privacy.
- This proof is made feasible by a secret key generated prior to the transaction taking place.
- The cryptocurrency Zcash uses Zk-SNARK as part of its protocol.

1.4 Features of zk-SNARKs ✓ ✓

1. **Zero-Knowledge:** The verifier learns nothing about the underlying data other than the fact that the prover is aware of it.
2. **Succinctness:** The proof size is quite small (usually a few hundred bytes) and does not depend on the size of the statements being proven. ✓
3. **Non-Interactive:** The proof is comprised of a single message from the prover to the verifier, eliminating the need for back-and-forth communication.
4. **Arguments of Knowledge:** The proof guarantees that the prover understands the information at hand and is not simply producing a legitimate proof by accident.

1.5 How zk-SNARKs Work

1. **Setup phase:** A trustworthy setup phase generates public parameters such as a proving key and a verification key. This phase entails producing structured reference strings (SRS), which are used to generate and verify proofs. ✓
2. **Proving Phase:** The prover generates a proof using the proving key and the witness (the hidden information). This proof persuades the verifier that the prover knows the witness without disclosing it. ✓
3. **Verification Phase:** The verifier checks the proof's validity using the verification key. The verification procedure is incredibly efficient, generally taking the same amount of time regardless of the magnitude of the underlying computation. ✓

- zk-SNARKs are based on polynomial equations and elliptic curves.
- zk-SNARKs are used in cryptocurrencies such as Zcash to provide private transactions, in which transaction data (such as sender, receiver, and amount) are hidden from the public while the transaction remains valid.
- A major drawback of ZK-SNARK implementation is the requirement for a trustworthy setup phase. During this phase, the Common Reference String is generated, which may jeopardize the system's security if not executed correctly.
- However, there are efforts to address this issue, such as using multiparty computation to divide confidence across different parties.

1.6 Features of zk-STARKs

1. **Zero-Knowledge:** The verifier learns nothing about the underlying data other than the fact that the prover is aware of it.
2. **Scalability:** zk-STARKs are intended to handle massive computations efficiently, producing proofs that stay manageable as the size of the data or calculation increases.
3. **Transparency:** The zk-STARKs eliminate the requirement for a trustworthy setup phase. They rely on publicly verifiable randomness, which increases trust and decreases the risk associated with the initial setup step in zk-SNARKs.
4. **Arguments of Knowledge:** Ensures that the prover understands the information rather than simply producing a valid proof by chance.

1.7 How zk-STARKs Work

1. **Setup phase:** Unlike zk-SNARKs, zk-STARKs don't need a trusted setup. Instead, they use publicly verifiable randomization to generate the parameters required for the proof system. The removal of the trusted setup phase reduces a substantial security risk, making zk-STARKs more safe and easier to deploy.
2. **Proving Phase:** The prover encodes the computation as a polynomial and employs interactive oracle proofs to demonstrate that the polynomial fulfills the required properties. This entails breaking down the computation into a series of steps that can be checked effectively.

1.8 How zk-STARKs Work

3. **Verification Phase:** The verifier checks the proof by sampling a few points of the polynomial and confirming that the sampled values meet the requirements. This enables the verifier to confirm the validity of the proof without having to perform the complete computation.

Zk starks work