**Boolean Access Control and Mathematical Proofs**

# Part 1: Boolean Logic for Identity Verification

Five Boolean variables represent user credentials and the access decision.

* **UserID**: true if the user ID is valid.
* **SS#**: true if the Social Security number is valid.
* **MothersName**: true if the mother’s maiden name is correct.
* **Password**: true if the password is valid.
* **Access**: true if at least three of the first four inputs are true.

All 16 combinations of the four-credential inputs are shown below; Access is true only when three or four of the inputs are true:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User ID** | **SS#** | **Mothers Name** | **Password** | **Access** |
| false | false | false | false | false |
| false | false | false | true | false |
| false | false | true | false | false |
| false | false | true | true | false |
| false | true | false | false | false |
| false | true | false | true | false |
| false | true | true | false | false |
| false | true | true | true | true |
| true | false | false | false | false |
| true | false | false | true | false |
| true | false | true | false | false |
| true | false | true | true | true |
| true | true | false | false | false |
| true | true | false | true | true |
| true | true | true | false | true |
| true | true | true | true | true |

Five combinations give Access=true (three or four inputs true). One Boolean formula capturing this rule is:

graphql

CopyEdit

Access = (UserID AND SS# AND MothersName)

OR (UserID AND SS# AND Password)

OR (UserID AND MothersName AND Password)

OR (SS# AND MothersName AND Password).

This sums all ways of choosing three credentials. A logic circuit implementing this rule could use four 3-input AND gates (each computing one of the above conjunctions) whose outputs feed a single 4-input OR gate. No NOT gates are required since we are only checking for true (correct) credentials.

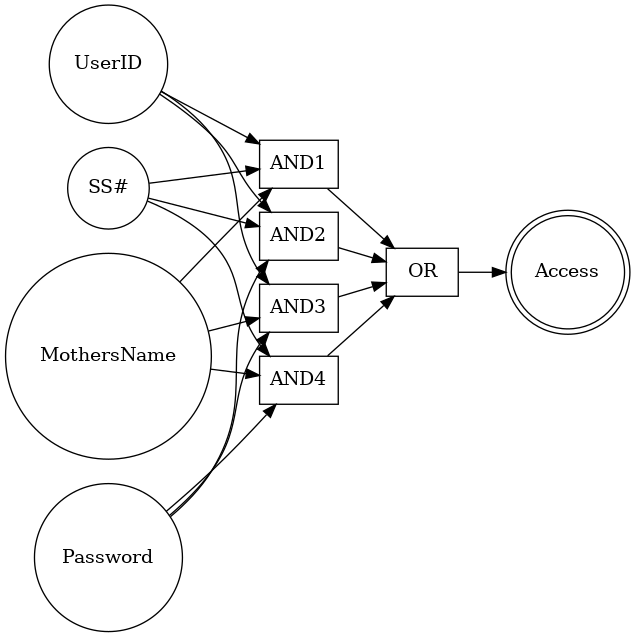


Figure 1. Boolean circuit diagram

**Part 2: Three Mathematical Proofs**  
**Proof 1: Sum of Integers (Induction)**  
We use induction.

* Base case: For holds.
* Inductive step: Assume. Then adding gives

Hence for all **.**

**Proof 2: Exponential Congruence (Counterexample)**  
The claim is false. For example, with one has versus, which are not equal. In general, equals (if ) or 0 (if ), while (if ) or 0 otherwise. These values differ for many choices of , so the claimed equality fails in general.

**Proof 3: Infinitely Many Primes (Contradiction)**  
Assume for contradiction there are only finitely many primes . Let . Dividing by any leaves remainder 1, so none of the listed primes divides. Thus must have a prime factor not in our list (or be prime itself), contradicting the assumption that we had all primes. Therefore, there must be infinitely many primes.