Identification Schemes and Entity Authentication\textsuperscript{1}

To prove your identity to someone, it is often said that you can use three things:

\begin{itemize}
  \item using what your are
  \item using what you have
  \item using what you know
\end{itemize}

For \textit{identification schemes}, the goal is to confirm someone’s identity based on what that person knows in \textit{real time}. (In contrast, with digital signatures, one can authenticate messages way after the messages have been signed.)

In practice, identification schemes must be simple enough so that it can be implemented in objects like smart cards and RFID’s. Its amount of communication and memory requirements must be small.

\textbf{The model.} In our model, Alice wishes to identify herself to Bob and maybe Bob to Alice as well. This will be done through an \textit{interactive protocol}. Alice and Bob will alternately talk to each other by sending information across a communication channel. Each run of the protocol is called a \textit{session}.

\textit{The adversary.} We will assume that the adversary can listen in to Alice and Bob’s conversation – i.e., s/he can see all the information that is being transmitted between the two of them.

\textit{Claim:} Identification schemes must involve randomness. That is, Alice cannot use the same information to verify her identity to Bob and someone else.

\textit{Why?}

\textsuperscript{1}This discussion is based on Chapter 9 of \textit{Cryptography: Theory and Practice}, 3rd Edition by Douglas Stinson.
Identification Schemes from Cryptographic Primitives

1. Challenge-and-response in the symmetric key setting

Review: Message Authentication Codes

Under this setting, we assume that Alice and Bob share a secret key $K$. Here’s one protocol they can use so Alice can confirm to Bob that she’s indeed Alice.

**Protocol 1a: Insecure Challenge-and-Response**

1. Bob chooses a random challenge $r$ which he sends to Alice.
2. Alice computes $y = MAC_K(r)$ and sends $y$ to Bob.
3. Bob computes $y' = MAC_K(r)$. If $y' = y$ then Bob accepts; otherwise, Bob rejects.
The problem: Protocol 1a is susceptible to a parallel session attack.

Protocol 1b: Secure Challenge-and-Response

1. Bob chooses a random challenge $r$ which he sends to Alice.
2. Alice computes $y = MAC_K(ID(Alice)||r)$ and sends $y$ to Bob.
3. Bob computes $y' = MAC_K(ID(Alice)||r)$. If $y' = y$ then Bob accepts; otherwise, Bob rejects.

Proof Sketch on why Protocol 1b is secure.
Mutual Authentication. This time around Alice and Bob need to prove their identities to each other. Consider the next protocol which “glues” together Protocol 1b:

**Protocol 2a: Insecure Mutual Challenge-and-Response**

1. Bob chooses a random challenge $r_1$ which he sends to Alice.

2. Alice chooses a random challenge $r_2$. She also computes $y_1 = MAC_K(ID(Alice)||r_1)$. She then sends $r_2$ and $y_1$ to Bob.

3. Bob computes $y'_1 = MAC_K(ID(Alice)||r_1)$. If $y'_1 = y_1$, Bob accepts; otherwise, Bob rejects. Bob also computes $y_2 = MAC_K(ID(Bob)||r_2)$ which he sends to Alice.

4. Alice computes $y'_2 = MAC_K(ID(Bob)||r_2)$. If $y'_2 = y_2$, Alice accepts; otherwise, Alice rejects.

The problem: Protocol 2a is susceptible to a *man-in-the-middle attack*.

**Protocol 2b: Secure Mutual Challenge-and-Response**

1. Bob chooses a random challenge $r_1$ which he sends to Alice.

2. Alice chooses a random challenge $r_2$. She then computes $y_1 = MAC_K(ID(Alice)||r_1||r_2)$. She then sends $r_2$ and $y_1$ to Bob.
3. Bob computes $y'_1 = MAC_K(ID(Alice)||r_1||r_2)$. If $y'_1 = y_1$, Bob accepts; otherwise, Bob rejects. Bob also computes $y_2 = MAC_K(ID(Bob)||r_2)$ which he sends to Alice.

4. Alice computes $y'_2 = MAC_K(ID(Bob)||r_2)$. If $y'_2 = y_2$, Alice accepts; otherwise, Alice rejects.

2. Challenge-and-response in the public key setting

In this setting, Alice and Bob no longer have a shared secret key. Instead, we assume that they both have a set of public and private keys that is used for a certain pre-specified digital signature scheme. We will also assume that each one of them has a certificate from a trusted authority (e.g., a certification authority) that "certifies" their public keys. The certificate for Alice is denoted as Cert(Alice) and contains $ID(Alice)$ and $p_{Alice}$, the public key of Alice. Likewise, Bob’s certificate, Cert(Bob), contains $ID(Bob)$ and $p_{Bob}$, the public key of Bob.

Below is a mutual identification scheme which is very similar to Protocol 2b. The main difference is the MAC in Protocol 2b is replaced with a digital signature scheme $sig$ whose verification algorithm is $ver$.

**Protocol 3: Public-key Mutual Identification Scheme**

1. Bob chooses a random challenge $r_1$. He sends Cert(Bob) and $r_1$ to Alice.

2. Alice chooses a random challenge $r_2$. She then signs the “message” $ID(Bob)||r_1||r_2$ using her private key to create the signature

$$y_1 = sig_{Alice}(ID(Bob)||r_1||r_2).$$

She sends Cert(Alice), $y_1$, $r_2$ to Bob.

3. Bob extracts Alice’s public key from Cert(Alice). He then checks that $ver_{Alice}(y_1) = (ID(Bob)||r_1||r_2)$. If yes, Bob accepts; otherwise, Bob rejects. Bob also signs the “message” $ID(Alice)||r_2$ using his private key to create the signature

$$y_2 = sig_{Bob}(ID(Alice)||r_2).$$
4. Alice extracts Bob’s public key from Cert(Bob). She checks that \( \text{ver}_{Bob}(y_2) = ID(Alice)||r_2 \). If yes, Alice accepts; otherwise, she rejects.

Identification Schemes from “Scratch”

1. The Schnorr Identification Scheme

Based on the Discrete Logarithm problem. We shall assume that there is a “network” of entities who need to verify their identities to each other using this scheme. There is a trusted authority who chooses a common set of system parameters:

- a large prime number \( p \) (with 1024 bits)
- a large prime divisor \( q \) of \( p - 1 \) (with 160 bits)
- a number \( \alpha \in Z_p^* \) whose order is \( q \) — i.e., \( \alpha^q \mod p = 1 \) and \( \alpha^k \mod p \neq 1 \) whenever \( 1 \leq k < q \).
- a security parameter \( t \) such that \( q > 2^t \). (The assumption here is the adversary’s probability of deceiving Alice or Bob will be \( 2^{-t} \). \( t = 40 \) is often adequate security for most practical applications.)

The parameters \( p, q, \alpha, t \) are all made public and used by everyone in the network. Everyone also chooses their own private and public keys as follows:

- **private key**: an integer \( a \), \( 0 \leq a \leq q - 1 \)
- **public key**: \( v = \alpha^{-a} \mod p = \alpha^{q-a} \mod p \).

The trusted authority then issues a certificate to everyone that includes their public keys.

**Protocol 4: Schnorr Identification Scheme**

1. Alice chooses a random number \( k \), where \( 0 \leq k \leq q - 1 \) and computes \( \gamma = \alpha^k \mod p \). She sends Cert(Alice) and \( \gamma \) to Bob.

2. Bob chooses a random challenge \( r \), \( 1 \leq r \leq 2^t \) and sends \( r \) to Alice.

3. Alice computes \( y = k + ar \mod q \), where \( a \) is her private key. She then sends the response \( y \) to Bob.

4. Bob extracts Alice’s public key \( v \) from Cert(Alice) and verifies that \( \gamma \cong \alpha^yv^r \mod p \). If so, Bob accepts; otherwise, Bob rejects.
Claim: Alice can always verify her identity to Bob.

Performance: