Hash Chains

\[ X_0 = \text{hash} \text{(random number)} \]
\[ X_1 = \text{hash}(X_0) \]
\[ X_2 = \text{hash}^2(X_0) \]
\[ X_3 = \text{hash}^3(X_0) \]
\[ \ldots \]
\[ X_n = \text{hash}^n(X_0) \]

An application of hash chains: Secure Multicast/Broadcast (Secure here is for integrity protection only)

The nodes authenticate the source by using digital signature

The source signs the message with his/her private key, \( K_{source} \{m\} \), before multicast/broadcast. All the recipients (assuming they know the public key of the source) can verify the integrity of the message.

**Problem:** What if they are lower-power nodes (sensors or IoT devices) that do not have the ability to perform public key functions?

**Solution:** TESLA method!

**TESLA**

\( X_n \) is somehow made available to everyone (possibly by one time signing the message with \( X_n \) or using another method)

Send \( m, \text{hash}(m, X_{n-1}) \) where \( X_{n-1} \) is the next element of the hash chain

Everyone knows \( m \) but NOT the key \( X_{n-1} \)

We send \((m, \text{hash}(m, X_{n-1}) )\) first and wait for some time before sending the key \( X_{n-1} \). Nodes receiving \( X_{n-1} \) can verify that this is indeed the element before \( X_n \) in the hash chain by hashing \( X_{n-1} \) to obtain \( X_n \). (Delayed Verification) Then the nodes can verify whether or not the integrity of message
m that they had received a little while ago is indeed preserved. \( X_{n-1} \) is discarded after one use. Next time, \( X_{n-2} \) is used by the source.

**Problem:** What if someone does not receive \( X_{n-1} \)?

**Solution:** If they receive \( X_{n-2} \) then they could generate \( X_{n-1} \) by using \( X_{n-2} \) and verify the message hashed with \( X_{n-1} \).

**Strong Password Protocol**

Most protocols are subject to offline dictionary attacks!

e.g. \( K_{AB}\{R\}, R \) \( \text{hash}(m, k), m \) etc used in many authentication protocol messages can be obtained by adversaries (possibly through eavesdropping)

**Properties of a Strong Password Protocol**

- Not subjected to an offline dictionary attack by an eavesdropper.
- Trudy cannot impersonate either Alice or Bob to obtain enough information for an offline dictionary attack.

**EKE – Encrypted Key Exchange**

An intruder would have to try all possible \( g^{ab} \mod p \) with every possible password \( w \! \)!

**Can Trudy impersonate Bob?**

- She chooses a ‘b’ and ‘c’
- She does NOT know ‘a’ therefore cannot generate \( K \)
- \( K \) is not generated using the password \( w \)

**Can Trudy impersonate Alice?**

- She could not create \( K\{c_1, c_2\} \) since there is no way she could obtain \( c_1 \)

**Problem:** This protocol is vulnerable to a server break-in (attacker can obtain ‘\( w \)’).

**Solution:** Store ‘\( w \)’ in some alternative form!

**Augmented Strong Password Protocols**
- If someone breaks into Bob, the attacker can only know $2^m \mod p$ (can do an offline dictionary attack but does not violate the strong password protocol requirements).
- Robust against eavesdroppers because they don’t know $a$, $b$, and $p$ to figure out $W$.
- It is important to ensure that, the value of $p$ is not known to Trudy (acting as Bob or Alice). Otherwise, she can try different values till a match is found.

**Credentials**
- Something that can be used to prove who you are or, to prove that you are authorized to perform some action.

(g,p) known to everyone

- This is a way of getting one's private key using a Strong Password Protocol.
- Only Alice knows $W$ and $W'$. Bob knows only $W$.
- Why $W$ and $W'$ are not same? If $W$ and $W'$ are same Bob can figure out Alice's private key.
- Trudy can’t impersonate either Alice or Bob to perform an offline dictionary attack.

**PFS, Escrow Foilage**

Note that the square brackets in messages in the protocol below is used for referring to digital signatures using Alice’s or Bob’s private key.
- Why add 1? We don’t want someone to replay the previous message.
- PFS? Yes, because the protocol uses DH values.
- No Person-in-the-middle attack because the protocol uses signed DH values.
- If Sam (Escrow, knows the private keys of Alice and Bob) is active and knows Alice’s and Bob’s private keys, the man in the middle attack is possible.