Tutorial 4

ID

1.1 Identification 1

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
	$\xrightarrow{(Alice,pw')}$	
		1. Alice is verified if $pw' = pw$

Fig. 1. The simplest Identification Protocol.

Question: explain why this protocol is bad.

1.2 Identification 2

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
1. Compute $H = H(pw)$		
	$\xrightarrow{(Alice, H)}$	
		2. Alice is verified if $H = H(pw)$

Fig. 2. The Identification Protocol Where $H(pw)$ is transmitted instead of pw.

Questions:
Is this protocol secure in terms of identification against the adversary?
Is this protocol more secure in terms of identification compared to the protocol 1?
What is the advantage of this protocol compared to the protocol 1?

1.3 Identification 3

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
1. Randomly choose $r \in \{0, 1\}^n$ 2. Compute $H = H(r, pw)$		
	$\xrightarrow{(Alice, r, H)}$	
		3. Alice is verified if $H = H(r, pw)$

Fig. 3. The Identification Protocol Using a Random Salt.

Questions:
Is this protocol secure in terms of identification against the adversary?
Is this protocol more secure in terms of identification compared to the protocol 2?
What is the advantage of this protocol compared to the protocol 2?

1.4 Identification 4

Alice : $(Alice, pw)$		**Bob** : $(Alice, pw)$
		1. Randomly choose $r \in \{0,1\}^n$
	$\xleftarrow{r}$	
2. Compute $H = H(r, pw)$		
	$\xrightarrow{(Alice, H)}$	
		3. Alice is verified if $H = H(r, pw)$

Fig. 4. The Identification Protocol Using a Random Salt Chosen by Bob.

Questions:
Is this protocol secure if the adversary cannot compute pw from the communication?
Is this protocol still secure if r chosen by Bob is always the same?

2.Key Exchange

Assume Alice has a (public,private) key pair (e,d). She wants to send her public key to Bob so they can establish a symmetric key based secure channel for sending the message M.

Is the following protocol for this secure (N is a nonce chosen by Alice)?

Alice → Bob: e,N
Bob → Alice: $E_e(k,N)$
Alice → Bob: $E_k(M)$

- Man-in-the-middle attack

Alice→Eve: e,N

Eve→ Bob: e’,N

Bob→Eve: Ee’(k,N)

Eve→ Alice: Ee(k’,N)

Alice→ Eve: Ek’(M)

Eve→ Bob: Ek(M)

So how can we fix it?

Have the public keys signed by a trusted certification authority. Assume the public key of Alice has been signed by a trusted certification authority CA and that the public key of CA is public so anyone (Bob in particular) can check the signature.

CertA = (Alice, e, expirydate,Sig_{CA})

Alice → Bob: CertA, N   Bob: Verifies e.
Bob → Alice: E_e(k, N)
Alice → Bob: E_k(M)

Is Bob sure that M was sent by Alice now?

Yes (assuming the M has some appropriate structure)! Although possibly no if it doesn’t. But wait …

3. Email Security

Sign Then Encrypt

- A has a pair of keys (d, e), where d is private and e is public
- B has a pair of keys (d’, e’), where d’ is private and e’ is public

  $A \rightarrow B: E_{e'}(A, M, Sign_d(M))$

- B believes that A sent the message, if the message and signature can be verified with e.
- A believes that only B can receive the signed M
- It provides authentication, non-repudiation, confidentiality and sender anonymity

Encrypt-then-Sign

- A has a pair of keys (d, e), where d is private and e is public
- B has a pair of keys (d’, e’), where d’ is private and e’ is public

  $A \rightarrow B: E_{e'}(A, M), Sign_{d}(E_{e'}(A, M))$

- Is there any problem with this approach?