$V'ig(k,m,(t_1,t_2)ig) = egin{cases} V(k,m,t_1) & ext{if } t_1 = t_2 \ ext{"0"} & ext{otherwise} \end{cases}$	
(i.e., $V'\left(k,m,\left(t_{1},t_{2} ight) ight)$ only outputs "1"	
a forger for (S',V') gives a forger for (S,V) .	
$S'(k,m) = S(k,m\oplus m)$ and $V'(k,m,t) = V(k,m\oplus m,t)$	
This should not be selected This construction is insecure because an adversary can	
request the tag for $m=0^n$ and thereby obtain a tag for any message. This follows from the fact that	
$m \oplus m = 0.$	
$S'(k,m)=S(k,m)[0,\dots,126]$ and $V'(k,m,t)=ig[V(k,m,t\ 0) ext{ or } V(k,m,t\ 1)ig]$	
(i.e., $V'(k,m,t)$ outputs ` 1" if either $t\ 0$ or $t\ 1$ is a valid tag for m)	
✓ Correct	
a forger for (S',V') gives a forger for (S,V) .	
$V'(k,m,t)=V(k,m\oplus 1^n,t).$	
\checkmark Correct a forger for (S',V') gives a forger for $(S,V).$	
$S'(k,m) = S(k,m)$ and $V(k,m,t) \text{if } m \neq 0^n$	
$V'(k,m,t) = \begin{cases} \ddots & \text{otherwise} \end{cases}$	
This should not be selected This construction is insecure because the adversary can simply output	
This should not be selected	
This construction is insecure because an adversary $ {\sf can \ request \ the \ tag \ for \ the \ message \ } 0^n \ {\sf and \ output \ the \ result \ as \ a \ valid \ forgery } $	
for the message 1^n .	
3. Recall that the ECBC-MAC uses a fixed IV (in the lecture we simply set the IV to 0). Suppose instead we chose a random IV for every message being signed and include the IV in the tag. In other words, $S(k,m) := \begin{pmatrix} r, & \mathrm{ECBC_r}\left(k,m\right) \end{pmatrix}$	1 / 1 point
where $\mathrm{ECBC}_r(k,m)$ refers to the ECBC function using r as the IV. The verification algorithm V given key k , message m ,	
and tag (r,t) outputs ``1" if $t=\mathrm{ECBC}_r\left(k,m ight)$ and outputs	
``O" otherwise. The resulting MAC system is insecure.	
An attacker can query for the tag of the 1-block message m and obtain the tag (r,t) . He can then generate the following	
existential forgery: (we assume that the underlying block cipher operates on n -bit blocks)	
$igodesize{}$ The tag $(r\oplus 1^n,\ t)$ is a valid tag for the 1-block message $m\oplus 1^n$.	
The tag $(r,t\oplus r)$ is a valid tag for the 1-block message 0^n . The tag $(r\oplus t,m)$ is a valid tag for the 1-block message 0^n .	
The tag $(r\oplus t,\ m)$ is a valid tag for the 1-block message 0^n . The tag $(m\oplus t,\ t)$ is a valid tag for the 1-block message 0^n .	
\checkmark Correct The CBC chain initiated with the IV $r\oplus m$ and applied	
to the message 0^n will produce exactly the same output as the CBC chain initiated with the IV r and applied to the	
message m . Therefore, the tag $(r\oplus 1^n,\ t)$ is a valid existential forgery for the message $m\oplus 1^n$.	
4. Suppose Alice is broadcasting packets to 6 recipients	0 / 1 point
B_1,\dots,B_6 . Privacy is not important but integrity is.	
packets he is receiving were sent by Alice.	
Alice decides to use a MAC. Suppose Alice and B_1,\dots,B_6 all share a secret key k . Alice computes a tag for every packet she	
sends using key k . Each user B_i verifies the tag when receiving the packet and drops the packet if the tag is invalid.	
Alice notices that this scheme is insecure because user B_1 can use the key k to send packets with a valid tag to	
users B_2, \dots, B_6 and they will all be fooled into thinking that these packets are from Alice.	
Instead, Alice sets up a set of 4 secret keys $S=\{k_1,\dots,k_4\}.$ She gives each user B_i some subset $S_i\subseteq S$	
of the keys. When Alice transmits a packet she appends 4 tags to it by computing the tag with each of her 4 keys. When user B_i receives	
by computing the tag with each of her 4 keys. When user D_i receives	
a packet he accepts it as valid only if all tags corresponding	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ $\checkmark \text{ Correct}$ Every user can only generate tags with the two keys he has.	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ Correct Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j , no user i	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ $\checkmark \text{Correct}$ $\text{Every user can only generate tags with the two keys he has.}$ $\text{Since no set } S_i \text{ is contained in another set } S_j, \text{ no user } i$ $\text{can fool a user } j \text{ into accepting a message sent by } i.$	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $ S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\} $ $ \text{Correct} $ $ \text{Every user can only generate tags with the two keys he has.} $ $ \text{Since no set } S_i \text{ is contained in another set } S_j, \text{ no user } i $ $ \text{can fool a user } j \text{ into accepting a message sent by } i. $ $ S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3,k_4\}, S_5 = \{k_2,k_3\}, S_6 = \{k_3,k_4\} $	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ $\checkmark \text{ Correct}$ $\text{Every user can only generate tags with the two keys he has.}$ $\text{Since no set } S_i \text{ is contained in another set } S_j, \text{ no user } i$ $\text{can fool a user } j \text{ into accepting a message sent by } i.$ $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3,k_4\}, S_5 = \{k_2,k_3\}, S_6 = \{k_3,k_4\}$ $! \text{ This should not be selected}$ $\text{User 4 can fool user 5 into believing that a packet}$ $\text{from user 4 was sent by Alice.}$ $S_1 = \{k_1,k_2\}, S_2 = \{k_2,k_3\}, S_3 = \{k_3,k_4\}, S_4 = \{k_1,k_3\}, S_5 = \{k_1,k_2\}, S_6 = \{k_1,k_4\}$	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ $\checkmark \text{ Correct}$ $\text{ Every user can only generate tags with the two keys he has.}$ $\text{ Since no set } S_i \text{ is contained in another set } S_j, \text{ no user } i$ $\text{ can fool a user } j \text{ into accepting a message sent by } i.$ $V \text{ Solid on the selected}$ $\text{ User 4 can fool user 5 into believing that a packet}$ $\text{ from user 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$ $V \text{ Solid on the selected}$ $\text{ User 4 was sent by Alice.}$	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_2, k_4\}, S_6 = \{k_3, k_4\}$ $\checkmark \text{Correct}$ $\text{Every user can only generate tags with the two keys he has.}$ $\text{Since no set } S_i \text{ is contained in another set } S_j, \text{ no user } i$ $\text{can fool a user } j \text{ into accepting a message sent by } i.$ $V S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3, k_4\}, S_5 = \{k_2, k_3\}, S_6 = \{k_3, k_4\}$ $V \text{In this should not be selected}$ $User 4 \text{ can fool user 5 into believing that a packet}$ $V \text{from user 4 was sent by Alice.}$ $V S_1 = \{k_1, k_2\}, S_2 = \{k_2, k_3\}, S_3 = \{k_3, k_4\}, S_4 = \{k_1, k_3\}, S_5 = \{k_1, k_2\}, S_6 = \{k_1, k_4\}$	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ $\checkmark \text{ Correct}$ $\text{Every user can only generate tags with the two keys he has.}$ $\text{Since no set } S_i \text{ is contained in another set } S_j, \text{ no user } i$ $\text{can fool a user } j \text{ into accepting a message sent by } i.$ $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3,k_4\}, S_5 = \{k_2,k_3\}, S_6 = \{k_3,k_4\}$ $! \text{This should not be selected}$ $\text{User 4 can fool user 5 into believing that a packet}$ $\text{from user 4 was sent by Alice.}$ $! \text{This should not be selected}$ $\text{User 5 can fool user 1 into believing that a packet}$ $\text{from user 5 was sent by Alice.}$	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_3,k_4\}$ $\checkmark \text{ Correct}$ Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j , no user i can fool a user j into accepting a message sent by i . $V S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3,k_4\}, S_5 = \{k_2,k_3\}, S_6 = \{k_3,k_4\}$! This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. $V S_1 = \{k_1,k_2\}, S_2 = \{k_2,k_3\}, S_3 = \{k_3,k_4\}, S_4 = \{k_1,k_3\}, S_5 = \{k_1,k_2\}, S_6 = \{k_1,k_4\}$! This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. $V S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3,k_4\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_3,k_4\}, S_6 = \{k_3,k_4\}$! This should not be selected User 5 was sent by Alice. $V S_1 = \{k_1,k_2\}, S_2 = \{k_1,k_3,k_4\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_3,k_4\}, S_6 = \{k_3,k_4\}$! This should not be selected	
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? \blacksquare $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_2, k_4\}, S_6 = \{k_3, k_4\}$ \checkmark Correct Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j , no user i can fool a user j into accepting a message sent by i . \blacksquare $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3, k_4\}, S_5 = \{k_2, k_3\}, S_6 = \{k_3, k_4\}$ \blacksquare This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. \blacksquare $S_1 = \{k_1, k_2\}, S_2 = \{k_2, k_3\}, S_3 = \{k_3, k_4\}, S_4 = \{k_1, k_3\}, S_5 = \{k_1, k_2\}, S_6 = \{k_1, k_4\}$ \blacksquare This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. \blacksquare $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3, k_4\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_2, k_3, k_4\}, S_6 = \{k_3, k_4\}$	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1, k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? ☑ $S_1 = \{k_1, k_2\}$, $S_2 = \{k_1, k_3\}$, $S_3 = \{k_1, k_4\}$, $S_4 = \{k_2, k_3\}$, $S_5 = \{k_2, k_4\}$, $S_6 = \{k_3, k_4\}$ ✓ Correct Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j , no user i can fool a user j into accepting a message sent by i . ☑ $S_1 = \{k_1, k_2\}$, $S_2 = \{k_1, k_3\}$, $S_3 = \{k_1, k_4\}$, $S_4 = \{k_2, k_3, k_4\}$, $S_5 = \{k_2, k_3\}$, $S_6 = \{k_3, k_4\}$! This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. ☑ $S_1 = \{k_1, k_2\}$, $S_2 = \{k_2, k_3\}$, $S_3 = \{k_3, k_4\}$, $S_4 = \{k_1, k_3\}$, $S_5 = \{k_1, k_2\}$, $S_6 = \{k_1, k_4\}$! This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. ☑ $S_1 = \{k_1, k_2\}$, $S_2 = \{k_1, k_3, k_4\}$, $S_3 = \{k_1, k_4\}$, $S_4 = \{k_2, k_3\}$, $S_5 = \{k_2, k_3, k_4\}$, $S_6 = \{k_3, k_4\}$! This should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice.	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_1 is given keys $\{k_1, k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_1 cannot validate the 3rd and 4th tags because he does not have k_2 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_2, k_4\}, S_6 = \{k_3, k_4\}$ \checkmark Correct Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j , no user i can fool a user j into accepting a message sent by i . $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3, k_4\}, S_5 = \{k_2, k_3\}, S_6 = \{k_3, k_4\}$! This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. $S_1 = \{k_1, k_2\}, S_2 = \{k_2, k_3\}, S_3 = \{k_3, k_4\}, S_4 = \{k_1, k_3\}, S_5 = \{k_1, k_2\}, S_6 = \{k_3, k_4\}$! This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3, k_4\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_2, k_3, k_4\}, S_6 = \{k_3, k_4\}$! This should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. Solution 1 into believing that a packet from user 5 was sent by Alice. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message m comprising of n AES blocks. Let m' be the n -block message obtained from m by flipping the	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> , is given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₁ cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₃ or <i>k</i> ₄ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? Solution of the selected solution of	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> , is given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₁ cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₃ or <i>k</i> ₄ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? Solution of the control of	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> , is given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₁ cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₃ or <i>k</i> ₄ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? Solution of the selection of the select	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_i is given keys $\{k_1,k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_i cannot validate the 3rd and 4th tags because he does not have k_3 or k_4 . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? $S_i = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_4\}, S_6 = \{k_5,k_4\}$ \checkmark correct Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_j , no user i can fool a user j into accepting a message sent by i . $S_i = \{k_1,k_2\}, S_2 = \{k_1,k_3\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3,k_4\}, S_5 = \{k_2,k_3\}, S_6 = \{k_3,k_4\}$ 1 This should not be selected User 4 can fool user S_i into believing that a packet from user 4 was sent by Alice. $S_i = \{k_1,k_2\}, S_2 = \{k_2,k_3\}, S_3 = \{k_3,k_4\}, S_4 = \{k_1,k_3\}, S_5 = \{k_1,k_2\}, S_6 = \{k_1,k_4\}$ 1 This should not be selected User 5 can fool user S_i into believing that a packet from user S_i was sent by Alice. $S_i = \{k_1,k_2\}, S_2 = \{k_1,k_3,k_4\}, S_3 = \{k_1,k_4\}, S_4 = \{k_2,k_3\}, S_5 = \{k_2,k_3,k_4\}, S_6 = \{k_3,k_4\}$ 1 This should not be selected User $S_i = \{k_1,k_2\}, S_i = \{k_1,k_3,k_4\}, S_i = \{k_1,k_4\}, S_i = \{k_2,k_3\}, S_i = \{k_2,k_3\}, S_i = \{k_3,k_4\}, S_i = \{k_1,k_4\}, S_$	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> _i are valid. For example, if user <i>B</i> _i is given keys { <i>k</i> _i , <i>k</i> _s } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> _i cannot validate the 3rd and 4th tags because he does not have <i>k</i> _s or <i>k</i> _s . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? Si = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₃ }, <i>S</i> ₁ = { <i>k</i> ₁ , <i>k</i> ₄ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₄ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ } ✓ correct Every user can only generate tags with the two keys he has. Since no set <i>S</i> ₁ is contained in another set <i>S</i> ₁ , no user <i>i</i> can fool a user <i>j</i> into accepting a message sent by <i>i</i> . Si = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₄ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ } ! This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. Si = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₄ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ }. ! This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. Si = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₄ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ }, I This should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message m comprising of n AES blocks. Let m' be the n-block message obtained from m by flipping the last bit of m' is b@ 1). How many calls to AES would it take to compute the tag for m' from the tag for m and the MAC key? (in this question please ignore message padding and simply assume that the m	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in S_i are valid. For example, if user B_i is given keys $\{k_1, k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that B_i cannot validate the 3rd and 4th tags because he does not have k_i as k_i . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? Si = {k_1, k_2}, S_2 = {k_1, k_3}, S_2 = {k_1, k_4}, S_4 = {k_2, k_3}, S_5 = {k_2, k_4}, S_4 = {k_3, k_4} Correct Every user can only generate tags with the two keys he has. Since no set S_i is contained in another set S_i , no user i can fool a user j into accepting a message sent by i . Si = {k_1, k_2}, S_2 = {k_1, k_3}, S_2 = {k_1, k_4}, S_4 = {k_2, k_3, k_4}, S_5 = {k_2, k_3}, S_4 = {k_3, k_4} I This should not be selected User 4 can fool user 15 into believing that a packet from user 4 was sent by Alice. Si = {k_1, k_2}, S_2 = {k_2, k_3, k_4}, S_3 = {k_3, k_4}, S_4 = {k_3, k_3}, S_5 = {k_1, k_3}, S_5 = {k_1, k_3}, S_6 = {k_3, k_4} I This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. Si = {k_1, k_2}, S_2 = {k_1, k_3, k_2}, S_2 = {k_1, k_3}, S_4 = {k_2, k_3}, S_5 = {k_2, k_3, k_4}, S_6 = {k_3, k_4} I This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. Si = {k_1, k_2}, S_2 = {k_1, k_3, k_2}, S_3 = {k_1, k_4}, S_4 = {k_2, k_3}, S_5 = {k_2, k_3, k_4}, S_6 = {k_3, k_4} I This should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for m and packet from user 5 was sent by Alice. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for m from the tag for m and the MAC key? (in this question please ignore message padding and simply assume that the message length is always a multip	
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> ₁ is given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₁ cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₁ as <i>k</i> ₂ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? S i = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₃ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₄ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ } ✓ Cerrect Every user can only generate tags with the two keys he has. Since no set <i>S</i> ₁ is contained in another set <i>S</i> ₂ , no user if can fool a user j into accepting a message sent by i. S i = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ } I this should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. S i = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₄ = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ }, . I this should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. S i = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ }. I this should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. 5. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message mecomprising of n AES blocks. Let m' be the hi-block message obtained from m by flipping the last bit of m' is 6 then the last bit of m is 6 then the last bit	1/1 point
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> ; is given keys (<i>k</i> ₁ , <i>k</i> ₂) he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ; cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₁ , or <i>k</i> ₂ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? Si = {k ₁ , k ₂ }, <i>S</i> ₂ = {k ₁ , k ₃ }, <i>S</i> ₁ = {k ₁ , k ₄ }, <i>S</i> ₄ = {k ₂ , k ₃ }, <i>S</i> ₅ = {k ₂ , k ₄ }, <i>S</i> ₄ = {k ₃ , k ₄ } Correct Every user can only generate tags with the two keys he has. Since no set <i>S</i> ₁ is contained in another set <i>S</i> ₂ , no user i can fool a user j into accepting a message sent by i. Si = {k ₁ , k ₂ }, <i>S</i> ₂ = {k ₁ , k ₃ }, <i>S</i> ₂ = {k ₁ , k ₄ }, <i>S</i> ₄ = {k ₂ , k ₃ , k ₄ }, <i>S</i> ₅ = {k ₂ , k ₂ }, <i>S</i> ₆ = {k ₃ , k ₄ } I this should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. Si = {k ₁ , k ₂ }, <i>S</i> ₂ = {k ₁ , k ₃ , k ₄ }, <i>S</i> ₃ = {k ₁ , k ₃ }, <i>S</i> ₄ = {k ₁ , k ₃ }, <i>S</i> ₅ = {k ₁ , k ₃ }, <i>S</i> ₆ = {k ₁ , k ₄ } I this should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. Si = {k ₁ , k ₂ }, <i>S</i> ₂ = {k ₁ , k ₃ , k ₄ }, <i>S</i> ₃ = {k ₁ , k ₃ }, <i>S</i> ₄ = {k ₂ , k ₃ }, <i>S</i> ₅ = {k ₂ , k ₃ , k ₄ }, <i>S</i> ₆ = {k ₃ , k ₄ } I this should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message mcomprising of n AES blocks. Let m' be the n-block message obtained from m by filipping the last bit of m is 5 then the tast bit of m is 6 then the tast bit of m is 6 then the last bit of m is 8 then the tast bit of m is 8 then the tast bit of m is 9 then the last bit of m is 9 then the last bit of m is 9 then the last bit of the result, and re-apply the two encryptions.	
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> ; is given keys (<i>k</i> ₁ , <i>k</i> ₂) he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ; cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₂ as <i>k</i> ₃ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? S : { {k}_1, k_2}, S_2 = {k}_1, k_3}, S_3 = {k}_1, k_3}, S_2 = {k}_2, k_3}, S_3 = {k}_2, k_4}, S_4 = {k}_3, k_4} ✓ Cerrect Every user can only generate tags with the two keys he has. Since no set <i>S</i> ; is contained in another set <i>S</i> ; no user <i>i</i> can fool a user <i>j</i> into accepting a message sent by <i>i</i> . S : { {k}_1, k_2}, S_2 = {k}_1, k_3}, S_3 = {k}_1, k_3}, S_2 = {k}_2, k_3, k_4}, S_5 = {k}_2, k_3}, S_6 = {k}_3, k_4} ! This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. S : { {k}_1, k_2}, S_2 = {k}_2, k_3}, S_3 = {k}_3, k_4}, S_4 = {k}_1, k_5}, S_6 = {k}_1, k_4} ! This should not be selected User 5 can fool user 1 into believing that a packet from user 3 was sent by Alice. S : { {k}_1, k_2}, S_2 = {k}_1, k_3, k_1}, S_3 = {k}_1, k_4}, S_6 = {k}_2, k_3, k_4} ! This should not be selected User 5 can fool user 1 into believing mat a packet from user 5 was sent by Alice. S : { {k}_1, k_2}, S_7 = {k}_1, k_2, k_1}, S_7 = {k}_1, k_2}, S_8 = {k}_2, k_3, k_3}, S_9 = {k}_2, k_3, k_3}, S_9 = {k}_2, k_3, k_3} • This should not be selected User 5 can fool user 4 into believing mat a packet from user 5 was sent by Alice. 5. Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message me comprising of n AES blocks. Let m' be the n-block message obtained from m by flipping the last bit of m' id.e. if the last bit of m' is 5 then the last bit of m' is 6 then the message length is	
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> ₁ is given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₁ cannet validate the 3rd and 4th tags because he does not have <i>k</i> ₂ not <i>k</i> ₃ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? S ₁ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₃ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₁ }, <i>S</i> ₄ = { <i>k</i> ₀ , <i>k</i> ₀ }, <i>S</i> ₅ = { <i>k</i> ₀ , <i>k</i> ₄ }, <i>S</i> ₆ = { <i>k</i> ₁ , <i>k</i> ₄ }, ✓ corvect Every user can only generate tags with the two keys he has. Since no set <i>S</i> ₁ is contained in another set <i>S</i> ₁ , no user if can fool a user j into accepting a mestage sent by i. S ₁ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₀ }, <i>S</i> ₁ = { <i>k</i> ₁ , <i>k</i> ₁ }, <i>S</i> ₁ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₄ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₆ = { <i>k</i> ₂ , <i>k</i> ₂ }. I has should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. S ₁ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₆ = { <i>k</i> ₁ , <i>k</i> ₂ }, I has should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. S ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>k</i> ₃ }. I has should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. S ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₇ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₈ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₈ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₈ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₈ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₈ = { <i>k</i> ₂ , <i>k</i> ₃ , <i>k</i> ₃ },	
a packet he accepts it as valid only if all tags corresponding to his keys in S. are valid. For example, if ture ZB, is given keys {k_i, k_j} he will accept an incoming packet only if the first and second lags are valid. Note that B, cannot validate the 3rd and 4th tags because he does not have k_g or k_i. How should alice assign keys to the 6 users so that no single user can forge packets on behalf of alice and fool some other user? S = {k_i, k_j}, S_2 = {k_i, k_g}, S_3 = {k_i, k_k}, S_4 = {k_g, k_g}, S_5 = {k_g, k_k}, S_6 = {k_g, k_k} ✓ correct Every user can only generate tags with the two keys he has. Since no set S_it is contained in another set S_j, no user if can fool a user j into accepting a message sent by it. S := {k_i, k_k}, S_2 = {k_i, k_g}, S_2 = {k_i, k_k}, S_4 = {k_g, k_i, k_k}, S_6 = {k_g, k_g}, S_6 = {k_g, k_k}, S_6 =	
a packet he accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> ₁ is given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₁ cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₂ or <i>k</i> ₃ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? S := { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i>	
a packet the accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>B</i> , it given keys { <i>k</i> ₁ , <i>k</i> ₂ } he will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ; cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₂ or <i>k</i> ₂ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? S ;= {k ₁ , k ₂ }, S ;= {k ₁ , k ₃ }, S ;= {k ₁ , k ₃ }, S ;= {k ₁ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₃ , k ₄ } Correct Every user can only generate tags with the two keys he hus. Since no set <i>S</i> ; is contained in another set <i>S</i> ; no user i can fool a user j into accepting a message sent by i. S ;= {k ₁ , k ₂ }, S ;= {k ₁ , k ₂ }, S ;= {k ₁ , k ₂ }, S ;= {k ₂ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₃ , k ₄ } ! This should not be selected User 4 can fool user 5 into believing that a packet from user 4 was sent by Alice. S ;= {k ₁ , k ₂ }, S ;= {k ₂ , k ₃ }, S ;= {k ₁ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₃ , k ₄ } ! This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. S ;= {k ₁ , k ₃ }, S ;= {k ₁ , k ₃ , k ₄ }, S ;= {k ₁ , k ₄ }, S ;= {k ₁ , k ₃ }, S ;= {k ₁ , k ₃ , k ₄ } ! This should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. S ;= {k ₁ , k ₃ }, S ;= {k ₁ , k ₃ , k ₄ }, S ;= {k ₁ , k ₃ }, S ;= {k ₂ , k ₃ }, S ;= {k ₃ , k ₄ }, S ;= {k ₃ , k ₃ }, S ;= {k ₃ ,	
a packet the accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. for example, if use <i>R</i> , it, is given keys { <i>k</i> , <i>k</i> ₀ } be will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ; cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₀ , or <i>k</i> ₀ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of alice and fool some other user? S ₁ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, S ₅ = { <i>k</i> ₂ , <i>k</i> ₄ }, S ₅ = { <i>k</i> ₁ , <i>k</i> ₂ }, ✓ correct Devy user can only generate tags with the two keys he has. Since no set <i>S</i> ₁ is contained in another set <i>S</i> ₂ , no user i can fool a user j into accepting a message sent by i. S ₁ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, S ₅ = { <i>k</i> ₂ , <i>k</i> ₃ }, S ₆ = { <i>k</i> ₃ , <i>k</i> ₄ }, 1 *This should not be selected User 4 can features 5 into believing that a packet from user 4 west sent by Alice. S ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, S ₂ = { <i>k</i> ₂ , <i>k</i> ₃ }, S ₃ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₄ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₆ = { <i>k</i> ₃ , <i>k</i> ₄ }, S ₇ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₇ = { <i>k</i> ₃ , <i>k</i> ₃ , S ₇ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₇ = { <i>k</i> ₃ , <i>k</i> ₄ }, 1 *This should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. S ₃ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₇ = { <i>k</i> ₃ , <i>k</i> ₃ , S ₇ = { <i>k</i> ₃ , <i>k</i> ₄ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₇ = { <i>k</i> ₃ , <i>k</i> ₄ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ , S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₄ }, 1 *This should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ , S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₃ }, S ₈ = { <i>k</i> ₃ , <i>k</i> ₄ }, 1 *This should not be selected This construction is not collision o	
a packet the accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if use <i>R</i> is, given keys (<i>k</i> ₁ , <i>k</i> ₂) be will accept an incoming packet only if the first and second tags are valid. Note that <i>B</i> ₂ cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₂ , or <i>k</i> ₁ . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? S ₁ = (<i>k</i> ₁ , <i>k</i> ₂), <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₁ , <i>k</i> ₂ }, ✓ currect Curry vaer can only generate tags with the two keys he has. Since no set <i>S</i> ₁ is contained in another set <i>S</i> ₂ , no user i can fool a user j into accepting a message sent by i. S ₁ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, If this should not be selected User 4 can fool user 5 into beliefying that a packet from user 4 was sent by Alice. S ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₄ }, S ₅ = { <i>k</i> ₃ , <i>k</i> ₃ , <i>S</i> ₅ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, If this should not be selected User 5 can fool user 1 into believing that a packet from user 5 was sent by Alice. S ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>k</i> ₃ , <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ , <i>k</i> ₃ }, If this should not be selected User 5 can fool user 4 into believing that a packet from user 5 was sent by Alice. S ₃ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ , <i>S</i> ₃ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₅ = { <i>k</i> ₃ , <i>k</i> ₃ }, If this should not be selected This construction is not collision from the same that the message length is always a multiplie of the Af5 blocks. Let m' be the n-block message obtained from m' by flipping the last bit of m' let 6 list bit of m' is 6 then the	
a packet the accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if use <i>R</i> is, if given keys { <i>k</i> , <i>k</i> , } he will accept an incoming packet only if the first and second regard are valid. Note that <i>B</i> , cannot validate the 2rd and 4th tags because he does not have <i>k</i> , or <i>k</i> _k . How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and does some other user? S is { {k}_1, k_2}, S = {k}_1, k_2}, S = {k}_1, k_2}, S = {k}_2, k_3}, S = {k}_2, k_3}, S = {k}_3, k_3}, S = {k}_4, k_4} ✓ correct Devy user can only generate tags with the two keys he has. Since no set <i>S</i> , is contained in another set <i>S</i> ₂ , no user i can fool a user j impo accepting a message sent by i. S : {k}_1, k_2}, S = {k}_2, k_3}, S = {k}_2, k_3}, S = {k}_2, k_3}, S = {k}_2, k_3}, S = {k}_3, k_3}, S = {k}_4, k_4} ! This anoth and be selected User 4 can fool user 8 into believing that a packet from user 4 was sent by Alice. S : {k}_1, k_2}, S = {k}_2, k_3}, S = {k}_3, k_4}, S = {k}_2, k_3}, S = {k}_3, k_4}, S = {k}_4, k_5} ! This should not be selected User 5 can fool user 8 into believing that a packet from user 6 was sent by Alice. S : {k}_3, k_5}, S = {k}_4, k_6, k_7}, S = {k}_1, k_4}, S = {k}_2, k_2}, S = {k}_3, k_4}, S = {k}_4, k_5}, S = {k}_5, k_6} ! This should not be selected User 5 can fool user 8 into believing that a packet from user 6 was sent by Alice. S : {k}_4, k_5}, S = {k}_6, k_6, k_7}, S = {k}_6, k_6}, S = {k}_6	
a packet the accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid for example, if view <i>P</i> , is, key his will accept an incoming packet only if the first and excent days are valid. Note that <i>B</i> ; cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₀ or <i>k</i> ₁ . How should alice assign keys to the 6 users so that no single user can forge packets only if the first and excent tags are valid. Note that <i>B</i> ; cannot validate the 3rd and 4th tags because he does not have <i>k</i> ₀ or <i>k</i> ₁ . Single (<i>k</i> ₁ , <i>k</i> ₂), <i>S</i> ₂ = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ₂ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₃ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₄ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₅ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ₆ = { <i>k</i> ₃ , <i>k</i> ₄ } \ ✓ correct bery user can only generate tags with the two legs he has. Since no set <i>S</i> ; is contained in another set <i>S</i> ₁ , no user <i>i</i> can fixe a user <i>j</i> into accepting a message set by <i>i</i> . Since no set <i>S</i> ; is expanded to the selected User <i>S</i> can fool user a limb believing that a packet from user 4 was sent by Alice. Since <i>S</i> ; = { <i>k</i> ₁ , <i>k</i> ₂ }, <i>S</i> ; = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₂ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃ , <i>k</i> ₃ }, <i>S</i> ; = { <i>k</i> ₃	
a packet the accepts it as valid only if all tags corresponding to his keys in <i>S</i> , are valid. For example, if user <i>D</i> is, given keys (<i>k</i> , <i>k</i> ₀) he will accept an incoming packet only if the first and second tags valid. Note that <i>D</i> is cannot validate the <i>B</i> id and 4th tags because he does not have <i>k</i> ₀ or <i>k</i> ₀ . How should alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and foot some other user? S = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₂ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₁ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₂ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₂ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₁ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₂ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i>S</i> ₃ = (<i>k</i> ₀ , <i>k</i> ₀), <i></i>	
a packet the accepts it as valid only if all tags corresponding to his keyn in <i>S</i> , are valid for example, if suce <i>D</i> , is given keys (<i>h</i> , <i>h</i> ₀) he will accept an incoming packet only if the first and second rays a valid. Note that <i>B</i> , connot validate the <i>B</i> rid and 4th tags because he does not have <i>k</i> ₀ are <i>k</i> ₀ . How should aftice assign keys to the 4 users so that no single user can forge packets on brhalf of Alice and foot some other user? S = (<i>h</i> , <i>h</i> ₀), <i>S</i> ₂ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₁ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₂ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₂ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i> ₃ = (<i>h</i> ₀ , <i>h</i> ₀), <i>S</i>	
a packet he accepts in as valid only if all tags corresponding to his keys in S. are valid. For example, if use \$P\$, it gives keys \$(k_1, k_2)\$ he will accept an incoming packet crushy if the first and second upon the Neet back \$P\$. Cannot validate the \$P\$ and 40 ht lags because he does not have \$k_1, or \$k_2\$. The value of the contract has been as the second of the contract of the packets on behalf of Alice and food some other user? S. \$= (k_1, k_2)\$. \$S_2 = (k_1, k_2)\$. \$S_4 = (k_1, k_2)\$. \$S_4 = (k_2, k_3)\$. \$S_4 = (k_3, k_3)\$. \$S_4 =	
a packet he accepts in a valid only if all tags corresponding to his keys in <i>S.</i> are valid. For example, if user <i>P.</i> it, given keys (<i>k</i> ₁ , <i>k</i> ₂) he will accept an incoming acceleration of the advention of the second value of the second val	
a packet he accepts in as valid only if all tags corresponding to his keys in S. are valid. For example, if use \$P\$, it gives keys \$(k_1, k_2)\$ he will accept an incoming packet crushy if the first and second upon the Neet back \$P\$. Cannot validate the \$P\$ and 40 ht lags because he does not have \$k_1, or \$k_2\$. The value of the contract has been as the second of the contract of the packets on behalf of Alice and food some other user? S. \$= (k_1, k_2)\$. \$S_2 = (k_1, k_2)\$. \$S_4 = (k_1, k_2)\$. \$S_4 = (k_2, k_3)\$. \$S_4 = (k_3, k_3)\$. \$S_4 =	
	So (k, m) = S(k, m em m) and V (k, m, c) → V(k, m em m, c) 1. This about muck a viscouse Throughout the controlled in

(i.e. output the first 32 bits of the hash)

This construction is not collision resistant

hash functions mapping inputs in a set M to $\{0,1\}^{256}$.

Our goal is to show that the function $H_{2}\left(H_{1}\left(m
ight)
ight)$ is also

suppose $H_{2}\left(H_{1}\left(\cdot
ight)
ight)$ is not collision resistant, that is, we are

collision resistant. We prove the contra-positive:

given x
eq y such that $H_2 \left(H_1 \left(x
ight) \right) = H_2 \left(H_1 \left(y
ight) \right)$.

This will prove that if H_1 and H_2 are collision resistant

then so is $H_{2}\left(H_{1}\left(\cdot\right)\right)$. Which of the following must be true:

We build a collision for either $H_{\mathbf{1}}$ or for $H_{\mathbf{2}}$.

x,y are a collision for H_2 .

 \bigcirc Either x,y are a collision for H_1 or

x,y are a collision for H_2 .

 \bigcirc Either x,y are a collision for H_2 or

✓ Correct

 $f_1(x,y) = \operatorname{AES}(y,x) \bigoplus y$,

Set $x_2=AES^{-1}(y_2,\ v\oplus y_1\oplus y_2)$

Set $x_2=AES^{-1}(y_2,\ v\oplus y_2)$

Set $x_2=AES^{-1}(y_2,\ v\oplus y_1)$

✓ Correct

You got it!

 $f_2(x,y) = AES(x,x) \bigoplus y.$

 $y_2=y_1\oplus AES(x_1,x_1)$

Set $y_2 = AES^{-1}(x_2,\ v \oplus y_1 \oplus x_2)$

 $H_{1}\left(x
ight) ,H_{1}\left(y
ight)$ are a collision for H_{1} .

If $H_{2}\left(H_{1}\left(x
ight)
ight)=H_{2}\left(H_{1}\left(y
ight)
ight)$ then

either $H_{\scriptscriptstyle 1}\left(x
ight)=H_{\scriptscriptstyle 1}\left(y
ight)$ and x
eq y , thereby giving us

 $H_{2}(H_{1}(x))=H_{2}(H_{1}(y))$ giving us a collision on H_{2} .

Either way we obtain a collision on $H_1\,$ or $H_2\,$ as required.

8. In this question you are asked to find a collision for the compression function:

Which of the following methods finds the required (x_1,y_1) and (x_2,y_2) ?

igotimes Choose x_1,y_1,y_2 arbitrarily (with $y_1
eq y_2$) and let $v := AES(y_1,x_1)$.

 \bigcirc Choose x_1,y_1,y_2 arbitrarily (with $y_1
eq y_2$) and let $v := AES(y_1,x_1)$.

 \bigcirc Choose x_1,y_1,y_2 arbitrarily (with $y_1
eq y_2$) and let $v := AES(y_1,x_1)$.

 \bigcirc Choose x_1,y_1,x_2 arbitrarily (with $x_1 \neq x_2$) and let $v := AES(y_1,x_1)$.

9. Repeat the previous question, but now to find a collision for the compression function

10. Let H:M o T be a random hash function where $|M|\gg |T|$ (i.e. the size of M is much larger

Which of the following methods finds the required (x_1,y_1) and (x_2,y_2) ?

igotimes Choose x_1, x_2, y_1 arbitrarily (with $x_1
eq x_2$) and set

 \bigcirc Choose x_1, x_2, y_1 arbitrarily (with $x_1
eq x_2$) and set

 \bigcirc Choose x_1, x_2, y_1 arbitrarily (with $x_1
eq x_2$) and set

 \bigcirc Choose x_1, x_2, y_1 arbitrarily (with $x_1
eq x_2$) and set

that finding a collision on H can be done with $Oig(|T|^{1/2}ig)$

in M such that H(x)=H(y)=H(z)?

random samples of $\boldsymbol{H}.$ How many random samples would it take

until we obtain a three way collision, namely distinct strings $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$

An informal argument for this is as follows: suppose we

samples is n choose 3 which is $O(n^3)$. For a particular

triple x,y,z to be a 3-way collision we need H(x)=H(y)

with probability $1/\lvert T \rvert$ (assuming H behaves like a random

function) the probability that a particular triple is a 3-way

collision is $O(1/|T|^2)$. Using the union bound, the probability

that some triple is a 3-way collision is $O(n^3/|T|^2)$ and since

we want this probability to be close to 1, the bound on \boldsymbol{n}

and H(x)=H(z). Since each one of these two events happens

collect n random samples. The number of triples among the n

 $y_2 = AES(x_1,x_1) \oplus AES(x_2,x_2)$

 $y_2=y_1\oplus x_1\oplus AES(x_2,x_2)$

Correct

than the size of T).

 $\bigcirc O(|T|^{2/3})$

 $O(|T|^{1/2})$

 $O(|T|^{1/3})$

✓ Correct

follows.

 $\bigcirc O(|T|)$

In lecture we showed

Awesome!

 $y_2 = y_1 \oplus AES(x_1, x_1) \oplus AES(x_2, x_2)$

Your goal is to find two distinct pairs (x_1,y_1) and (x_2,y_2) such that $f_1(x_1,y_1)=f_1(x_2,y_2)$.

a collision on H_1 . Or $H_1\left(x
ight)
eq H_1\left(y
ight)$ but

where $\ensuremath{\mathrm{AES}}(x,y)$ is the AES-128 encryption of y under key x.

 $H_{1}\left(x
ight) ,H_{1}\left(y
ight)$ are a collision for H_{2} .

 \bigcirc Either $H_2(x), H_2(y)$ are a collision for H_1 or

because an attacker can find a collision in time $2^{16}\,$ using

1/1 point

1/1 point

1/1 point

1/1 point

This should not be selected

the birthday paradox.

7. Suppose H_1 and H_2 are collision resistant

Week 3 - Problem Set

! Try again once you are ready

What tampering attacks are not prevented by this system?

1. Suppose a MAC system (S,V) is used to protect files in a file system by appending a MAC tag to

each file. The MAC signing algorithm ${\cal S}$ is applied to the file contents and nothing else.

Replacing the tag and contents of one file with the tag and contents of a file

Both files contain a valid tag and will be accepted at verification

2. Let (S,V) be a secure MAC defined over (K,M,T) where $M=\{0,1\}^n$ and $T=\{0,1\}^{128}$. That is, the key space is K, message space is $\{0,1\}^n$, and tag space is $\{0,1\}^{128}$.

Which of the following is a secure MAC: (as usual, we use | to denote string concatenation)

from another computer protected by the same MAC system, but a different key.

Week 3 - Problem Set

Swapping two files in the file system.

Erasing the last byte of the file contents.

Changing the first byte of the file contents.

TO PASS 80% or higher

LATEST SUBMISSION GRADE

✓ Correct

time.

70%

Graded Quiz • 20 min

Due Nov 18, 1:29 PM IST

GRADE 70%

1/1 point

Retake the assignment in 7h 35m