CS 395T

Formal Model for Secure Key Exchange

Main Idea: Compositionality

- Protocols don’t run in a vacuum
  - Security protocols are typically used as building blocks in a larger secure system
  - For example, a key exchange protocol such as IKE can be used to implement secure sessions
- A protocol can be “correct” when used in standalone mode, but completely broken when used as a building block in a larger system
- Objective: modular, composable definitions of protocol security

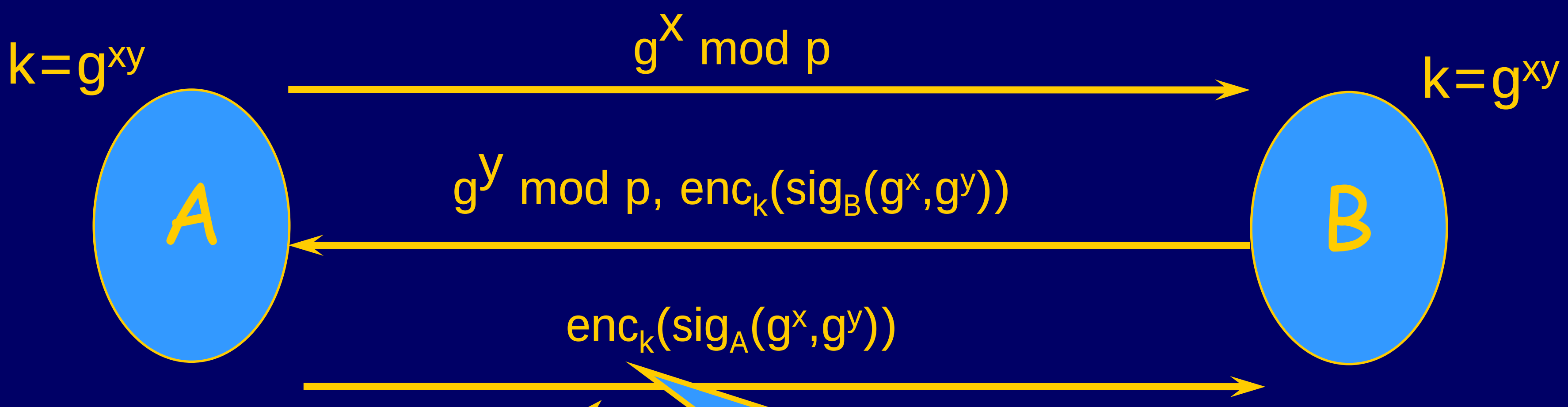
"Compositional" Definition of Security

Read and understand this paper! [Shoup '99]

- Definition should describe guarantees provided by a key exchange protocol to higher-level protocols
  - Security properties must hold in <u>any</u> environment in which the key exchange protocol is used
- Different types of attack
  - <u>Static corruptions</u>: adversary may operate under aliases, but honest users remain honest
  - <u>Adaptive corruptions</u>: adversary corrupts honest users
    - Learns either the long-term secret, or all of user's internal data
- Support anonymous (password-only) users

Station-to-Station Protocol

[Diffie et al. ’92]

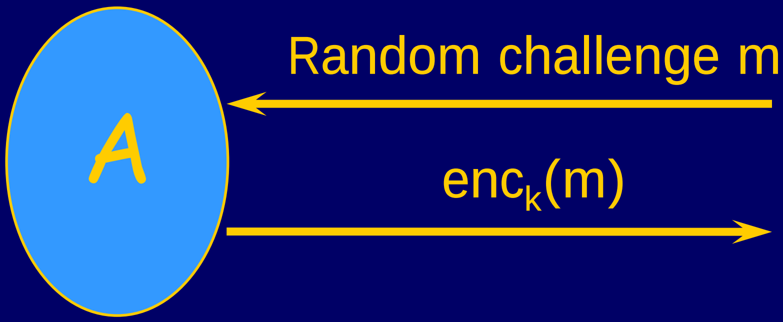


<u>Interleaving attack</u>:
Adversary replays B’s own encryption back to B.
Result: B thinks he is talking to himself, A thinks he is talking to B.

This encryption is critical.
Without it, adversary can send $sig_C(g^x,g^y)$.
Result: B thinks he is talking to C, while sharing a key with A, who thinks he is talking to B.

Protocol Interference Attack

- What if, in addition to STS, A executes some protocol where this interaction takes place:



Adversary picks $sig_C(g^x,g^y)$ as m, and learns $enc_k(sig_C(g^x,g^y))$, which he passes to B in the STS protocol, convincing B that B is talking to C.

- Problem: challenge-response protocols may be used as encryption oracles by the adversary
- Problem: “hijacking” of honest user’s public key
  - Fool CA into binding A’s public key to a different identity

Simulatability

- Security is indistinguishability between the ideal world and the real world
- In the ideal world, the protocol is secure by design
  - Key are distributed “magically” (no communication)
- The real protocol is secure if it can be simulated in the ideal world
  - Intuitively, this means that the real protocol leaks no more information than a protocol with “magic” channels
  - Transcripts (records of everything that happened) should be indistinguishable between the real world and a simulation in the ideal world

Ideal World

- Adversary creates and connects user instances
  - Users magically obtain secret keys that are hidden from the adversary
  - If users are "connected," then they magically share a common key which is hidden from the adversary
  - Adversary does not learn any information about the keys <u>except what is leaked through their use</u>
- No cryptography, no certificates, no messages
- Pure abstraction of the service that key exchange protocol provides to higher-level protocols

Adversary and Ring Master

- Define a game between the adversary and the "ring master"
  - Ring master generates random values and executes operations issued by the adversary
  - Interaction between the ring master and the adversary models permissible information leakage from protocol
- Operations allow the adversary to set up a secure session in the ideal world
  - Create users and user instances
  - Create and abort sessions between user instances
  - Apply a function

Ideal World: User Instances

- InitializeUser(i, ID_i)
  - Assign unique identity (bit string) ID_i to the ith user
  - User in the ideal world is simply a placeholder; he does not actually do anything
    - – Recall that session keys will be created magically
- InitializeUserInstance(i, j, $role_{ij}$, PID_{ij})
  - Same user may participate in multiple instances of the same protocol
  - I_{ij} is the identity of the new instance, which is communicating with some counterparty PID_{ij}
  - $role_{ij}$ is either 0, or 1

Adversary creates an instance of user i who will be talking to an instance of user j.

Ideal World: Session Key Generation

- StartSession(i, j, , [adversaryKey])
  - <u>Create</u>: ring master generates K_{ij} as random bit string
  - <u>Connect</u>(i',j'): ring master sets K_{ij} equal to $K_{i'j'}$
  - <u>Compromise</u>: ring master sets K_{ij} equal to adversaryKey

  Only permitted if PID_{ij} hasn't been assigned (static corruptions only)

  – "Create" models creation of a brand-new session key to be used between the ith and jth user

  – "Connect" models establishment of this session (the key magically becomes known to both user instances)

  – "Compromise" models adversary's corruption of a user

  - Add StartSession(i,j) to the transcript
- AbortSession(i, j)
  - This models failed attempt to establish a session

Ideal World: Information Leakage

- Application(f)
  - Ring master gives to the adversary $f(R,\{K_{ij}\})$ where R is some known random bit string and $\{K_{ij}\}$ is the set of all session keys. This is recorded in transcript.
    - f is a function or a program, may have side effects
  - Intuitively, function f defines what adversary may be able to learn after shared key has been established
    - For example, he may able to learn ciphertexts computed by user using some randomized symmetric cipher and the new key. We encode this cipher as function f.
- Implementation(comment)
  - Adversary puts arbitrary bit string into transcript

Real World: Registering Identities

- InitializeUser(i, ID_i)
  - Assign unique identity (bit string) ID_i to the ith user
  - User registers his identity with trusted third party T via protocol-specific, probabilistic registration routine
    - – This models issuance of a public-key certificate
- Register(ID, registrationRequest)

Not in the ideal world!

Not in the ideal world!

  - Adversary runs registration protocol directly with T, using protocol-specific registrationRequest bitstring
    - – This models PKI attacks: adversary obtains a certificate for an identity of his choice
  - The sets of identities in InitializeUser and Register must be disjoint

Real World: User Instances

- InitializeUserInstance(i, j, role$_{ij}$, PID$_{ij}$)
  - I_{ij} is the identity of the new instance, which is communicating with some counterparty PID$_{ij}$
  - role$_{ij}$ is either 0, or 1
  - User instance is a probabilistic state machine. Upon delivery of a message, it updates its state, generates a response message and reports status:
    - Continue: prepared to receive another message
    - Accept: finished & has generated session key K_{ij}
    - Reject: finished & refuses to generate session key

In the ideal world, user instances are placeholders

Real World: Messages

- DeliverMessage(i, j, inMsg)
  - Adversary delivers message inMsg to user instance I_{ij}
  - User instance updates its state, generates response message outMsg and reports its status
    - This models active interaction between the adversary controlling the network and the user in actual protocol
  - The following is recorded in the transcript:
    - Implementation(DeliverMessage,i,j,inMsg,outMsg,status)
    - StartSession(i,j) if status=accept
    - AbortSession(i,j) if status=reject
- DeliverMessageToTTP(inMsg)
  - Adversary delivers inMsg to T and receives outMsg

This is used to simulate real-world messages in ideal world (no messages in ideal world!)

Real World: Higher-Level Protocols

- Application(f)
  - Same in the real world as in the ideal world, except that $f(R,\{K_{ij}\})$ is computed as a function of actual session keys $\{K_{ij}\}$ and random input R
    - R is independent of any randomness used in initialization of user instances and protocol execution
  - As in the ideal world, add Application(f,$f(R,\{K_{ij}\})$) to the transcript

Definition of Security

- Termination
  - Any real-world user instances terminates after a receiving a polynomially-bounded number of messages
- Liveness
  - If adversary faithfully delivers msgs between two real-world user instances, they accept & share same key
- Simulatability
  - For any efficient real-world adversary A, there exists an efficient ideal-world adversary A’ such that transcripts RealWorld(A) and IdealWorld(A’) are computationally indistinguishable

Discussion

- Compositionality is much more than key secrecy
  - Application operation allows keys to be used in an <u>arbitrary</u> way by higher-level protocols
    - Can encode any higher-level key-based functionality as some function f, and then add Application(f) to the transcript
  - The real protocol is indistinguishable from the ideal functionality regardless of how keys are used later on
    - Can use key exchange protocol as it were perfectly secure
- Adversary's freedom to set up connections in the ideal world is illusory
  - To simulate a secure real-world protocol, connections will have to be set up in a very specific way

Simple Exercise

- Prove that protocol cannot satisfy simulatability if real-world adversary A can output session key after it has been established, but not yet used
  - Using constant f in Application, A records his guess of the key in the transcript. Using identity as f in Application, he has ring master record the actual key.
  - In real world, they are always equal
    - By assumption, A knows the key in the real world
  - In ideal world, they are equal with negligible probability
    - Key is generated randomly by ring master
  - This immediately gives a statistical test to distinguish the real and the ideal world

Crypto Review: DDH Assumption

- Let's review some crypto

  G is a group of large prime order q

  For $g_1, g_2, u_1, u_2 \in G$ define

$$DHP(g_1,g_2,u_1,u_2) = \begin{cases} 1 & \text{if } \exists x \in Z_q \text{ s.t. } u_1 = g_1^x,\ u_2 = g_2^x \\ 0 & \text{otherwise} \end{cases}$$

- Decisional Diffie-Hellman (DDH) Assumption says that there exists no efficient algorithm for computing DHP correctly with negligible error probability on all inputs

More DDH

- The following is implied by the DDH Assumption:
  Distributions
  $g, \{g^{x_i}\}, \{g^{y_j}\}, \{g^{x_i y_j}\}$ and
  $g, \{g^{x_i}\}, \{g^{y_j}\}, \{g^{z_{ij}}\}$ where $1 \le i \le n, 1 \le j \le m$,
  g, x_i, y_j, z_{ij} random
  are computationally indistinguishable
- DDH and Leftover Hash Lemma imply that the following are computationally indistinguishable:
  $g, g^x, g^y, k, H_k(g^{xy})$ and
  g, g^x, g^y, k, K where K is random bit string

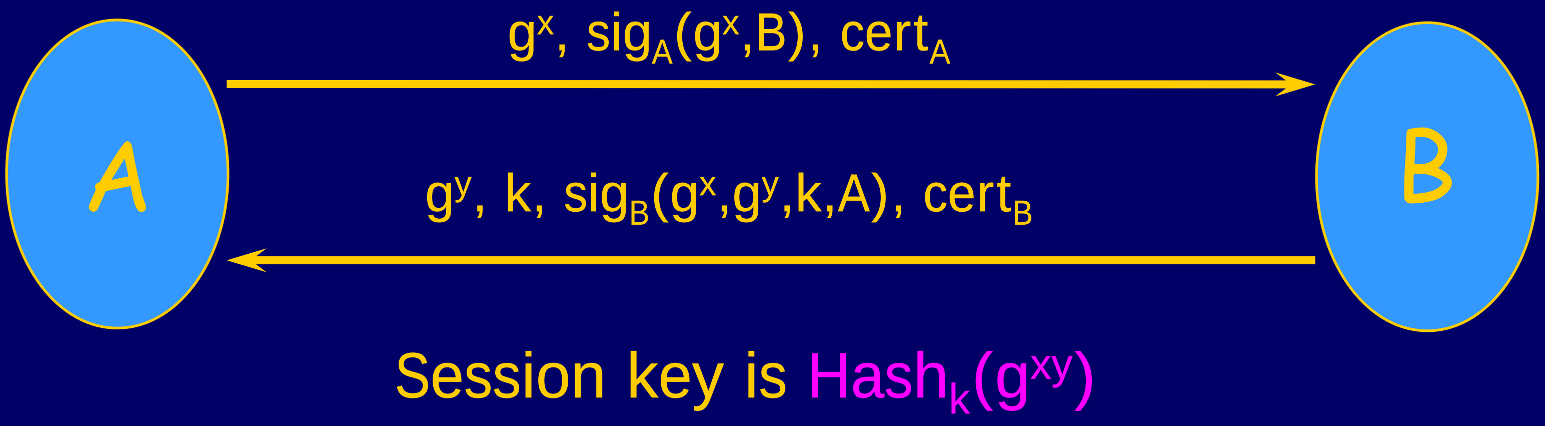
Security of Digital Signatures

It is infeasible for adversary to win following game:

1. Signing key is generated and given to the signing oracle. The corresponding public key is given to the adversary.
2. Adversary requests signatures on any messages of his choice. Messages may depend on received signatures.
3. Adversary wins the game if he outputs a message other than those on which he previously requested signatures along with a valid signature on that message.

This is known as security against existential forgery

DHKE Protocol


A
B
g^x, $sig_A(g^x,B)$, $cert_A$
g^y, k, $sig_B(g^x,g^y,k,A)$, $cert_B$

Session key is $Hash_k(g^{xy})$

Assuming the digital signature scheme is secure against existential forgery,

DHKE is a secure key exchange protocol under the DDH assumption

Proof of Simulatability

- Given real-world adversary A, construct ideal-world adversary A’ who simulates protocol execution to A
  - A should not be able to tell whether he is in the real or ideal (perfectly secure) world
    - Prove that no probab. poly-time test can distinguish transcript of real-world protocol execution from a simulation created by A’
- Basic idea: A’ runs A as a subroutine
  - When a session is established in the real world, A’ “connects” corresponding user instances in ideal world
  - Ring master in the ideal world substitutes real-world session keys with randomly generated ideal keys
- Must prove that key substitutions are undetectable

DHKE: Security for Responder (1)

- Suppose user instance B received 1st message and accepted
- If PID_B is not assigned to user, then "compromise" B in the ideal world
  - PID_B is initiator's identity (in responder's view)
    - PID_B not assigned means that the protocol is being executed with the adversary (or adversary-controlled user) as initiator
  - Extract session key from the responder in the real world, and use it as argument to the "compromise" operation in the ideal world

DHKE: Security for Responder (2)

- If PID_B has been assigned to user, then "create" B in the ideal world
  - This means that the protocol is being executed with an honest user as the initiator
  - "Create" models key creation in ideal world. Ring master creates a <u>random</u> session key for B. In the real world, the key is not random. It is computed as $Hash_k(g^{xy})$.
- DDH Assumption and Leftover Hash Lemma imply that $Hash_k(g^{xy})$ is computationally indistinguishable from a random key even if g^x, g^y, and k are known

DHKE: Security for Responder (3)

- The indistinguishability argument only works if A has not been, nor ever will be compromised
- Fortunately, “compromised” connections are not allowed if PID_A has been assigned
  - PID_A is responder’s identity (in initiator’s view). Because initiator sent 1st message to B, this means that PID_A=B and “compromise” is not allowed.
  - Intuition: corruptions are static. Once honest A starts talking to B, he cannot be compromised.

DHKE: Security for Initiator (1)

- Suppose user instance A received 2nd message and accepted
- If PID_A is not assigned to any user, then "compromise" A in the ideal world
  - PID_A is responder's identity (in initiator's view)
    - PID_A not assigned means that the protocol is being executed with the adversary (or adversary-controlled user) as responder
  - Extract session key from the responder in the real world, and use it as argument to the "compromise" operation in the ideal world

DHKE: Security for Initiator (2)

- If PID_A has been assigned to user B, then "connect" A and B in the ideal world
  - Protocol is being executed with honest responder B
  - "Connect" magically gives B's random session key to A
- Security of digital signature scheme guarantees that A's and B's values of g^x, g^y, and k match
  - Therefore, A's and B's keys are equal in the real world
- There is no detectable difference between worlds
  - A's and B's keys are equal in both worlds
  - In ideal world, keys are random. In real world, they are DH values, but this is not computationally detectable

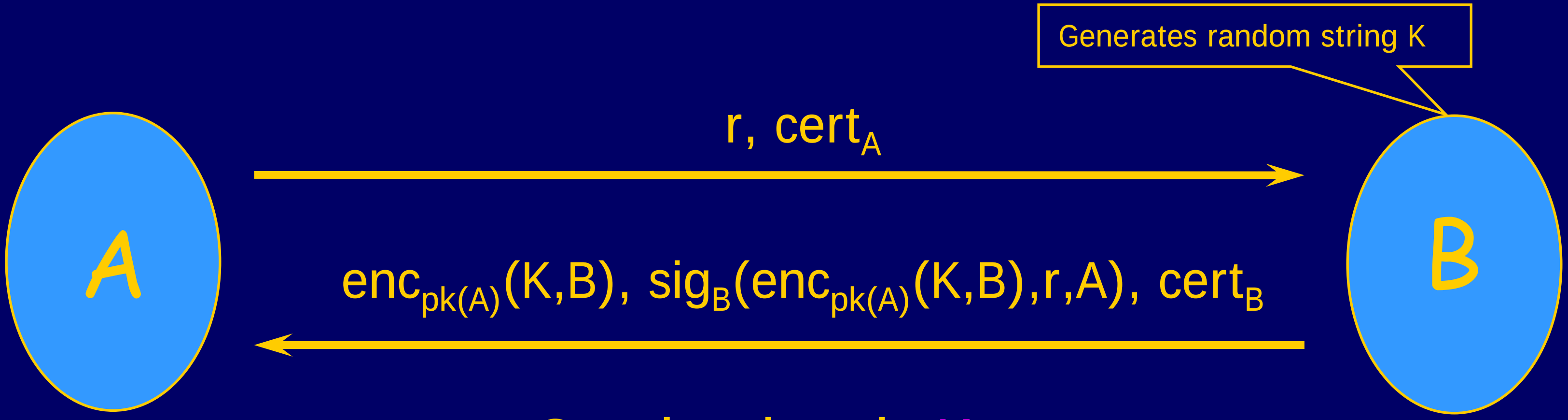
Crypto Review: Non-Malleability

Same as CCA-2 indistinguishability, i.e., adversary's probability of winning following game is close to ½:

1. Adversary requests encryption of any plaintext and/or decryption of any ciphertext
2. Adversary picks two plaintexts m_0 and m_1, and receives an encryption of m_b (b is a randomly chosen bit)
3. Adversary requests encryption of any plaintext and/or decryption of any ciphertext except that returned in step 2
4. Adversary wins if he outputs b'=b

Non-malleability says adversary can neither learn plaintext, nor modify it

EKE Protocol


Generates random string K
r, cert_A
A
B
$\text{enc}_{pk(A)}(K,B)$, $\text{sig}_B(\text{enc}_{pk(A)}(K,B),r,A)$, cert_B

Session key is K

Assuming the digital signature scheme is secure against existential forgery and the public-key encryption scheme is non-malleable,

EKE is a secure key exchange protocol

EKE: Security for Responder (1)

- Suppose user instance B received 1st message and accepted
- If PID_B is not assigned to user, then "compromise" B in the ideal world
  - PID_B is initiator's identity (in responder's view)
    - PID_B not assigned means that the protocol is being executed with the adversary (or adversary-controlled user) as initiator
  - Extract session key from the responder in the real world, and use it as argument to the "compromise" operation in the ideal world

EKE: Security for Responder (2)

- If PID_B has been assigned to user, then “create” B in the ideal world
  - This means that the protocol is being executed with an honest user as the initiator
  - In the ideal world, ring master creates a random session key for B. This key is not equal to the key that B sent to A under encryption in the real world.
- Real-world adversary cannot tell the difference between a random key generated by ring master and the key that B sent under encryption
  - Holds only if ciphertext sent by B is never decrypted
  - Proving this will rely on non-malleability

EKE: Security for Initiator (1)

- Suppose user A received 2nd message, but PID_A has not been assigned to a user
- If $enc_{pk(A)}$(K,B’) was generated by B’, A rejects
  - Some honest user B’ thinks he is responding to A, but his identity doesn’t match identity PID_A expected by A
  - Rejection does not require decrypting $enc_{pk(A)}$(K,B’)
    - This is important! Otherwise, ideal-world adversary would not know when to tell the ideal-world instance of A to reject.
- If $enc_{pk(A)}$(K,B’) not from another user, let A run. If A accepts, “compromise” A and extract key.
  - Requires decrypting $enc_{pk(A)}$(K,B’). This is Ok, since it was not created by an honest user.

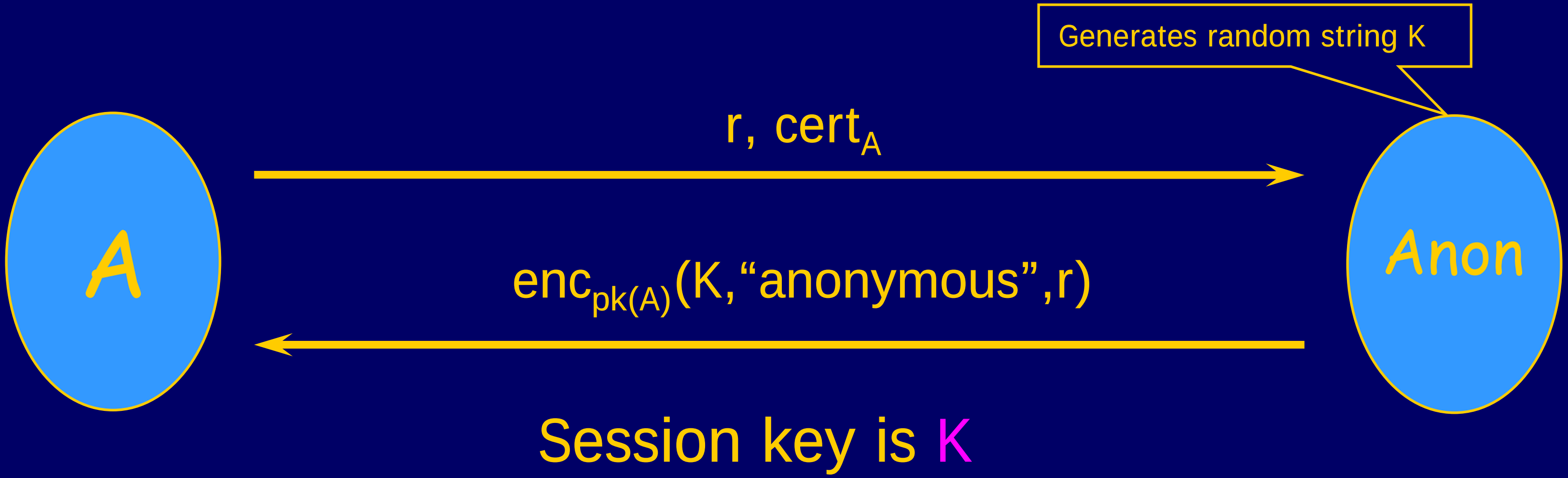
EKE: Security for Initiator (2)

- If PID_A has been assigned to user B, then "connect" A and B in the ideal world
  - If signature verifies correctly, then $enc_{pk(A)}(K,B)$ must have been created by B who thinks he is talking to A
    - Recall that A's identity is signed by B
  - No need to decrypt the ciphertext
  - Connection is valid because random values r are unique
- There is no detectable difference between worlds
  - A's and B's keys are equal in both worlds

Anonymous Users

- Add user with special "anonymous" identity
  - Treat all anonymous users as a single user
- StartSession(i, j, Compromise, adversaryKey) is legal if PID_{ij}="anonymous"
  - An anonymous user may be the adversary himself, so permit adversary to compromise anonymous users
- Simulatability-based definition of security naturally supports password protocols
  - Use secure key exchange to establish a secure channel, then authenticate with password on this channel
  - Like any other functionality based on secure sessions

A-EKE Protocol


Generates random string K
r, cert_A
A
Anon
$\text{enc}_{pk(A)}(K,\text{“anonymous”},r)$
Session key is K

Assuming the public-key encryption scheme is non-malleable,

EKE is a secure key exchange protocol

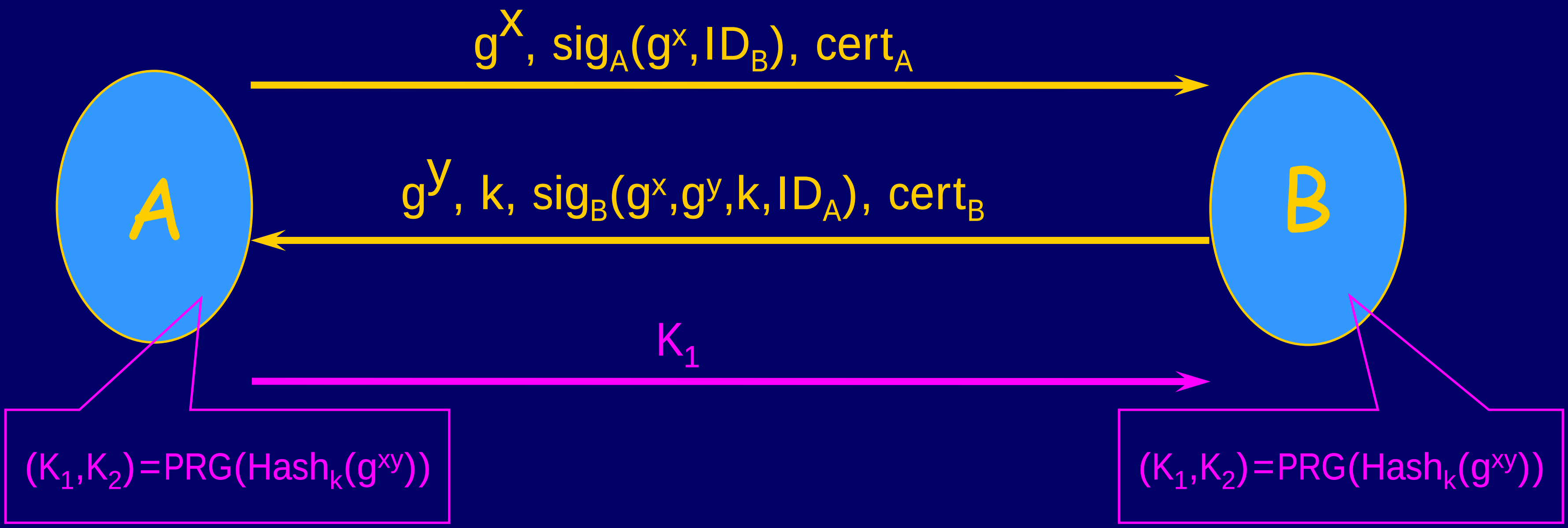
A-EKE: Security for Initiator

- Suppose PID_A=“anonymous”
  - Initiator thinks he is talking to the anonymous user
- If $enc_{pk(A)}$(K,“anonymous”,r) was <u>not</u> generated by “anonymous”, let A run. If A accepts, compromise A and extract key.
  - Requires decrypting $enc_{pk(A)}$(...). This is Ok, since it was not created by an honest user with identity.
- If $enc_{pk(A)}$(K,“anonymous”,r) was generated by “anonymous” and r received by “anonymous” matches r sent by A, connect.
- If r does not match, reject.

Adaptive Corruptions

- Users may be corrupted during protocol execution
  - Adversary learns user's long-term secret (private key)
  - Strong adaptive corruptions: learns user's entire state
- CorruptUser(i) operation in the ideal world
  - Gives no information to ideal-world adversary
- StartSession(i, j, Compromise, ...) is legal if PID_{ij} is assigned to corrupt user or user U_i is corrupt
  - Compromise is now allowed if either party in the protocol session is corrupt
- Neither DHKE, nor EKE is secure against adaptive corruptions

Security Against Adaptive Corruptions


g^x, $sig_A(g^x, ID_B)$, $cert_A$
A
B
g^y, k, $sig_B(g^x, g^y, k, ID_A)$, $cert_B$
K_1
$(K_1,K_2)=PRG(Hash_k(g^{xy}))$
$(K_1,K_2)=PRG(Hash_k(g^{xy}))$

Session key is K_2

This protocol provides <u>key confirmation</u>

(B doesn't start using the key until he receives confirmation that A is using the same key)