

Summer School on
Applied Cryptographic Protocols

*« Protocols for
Securing Communication »*

30 September 2010

Michel Abdalla
École normale supérieure & CNRS

-
- **A fundamental problem in cryptography**
 - Enable secure communication over insecure channels
 - **A common scenario**
 - Users encrypt and authenticate their messages using a shared secret key
 - **The secret key is usually obtained via a key exchange protocol**

Diffie-Hellman protocol

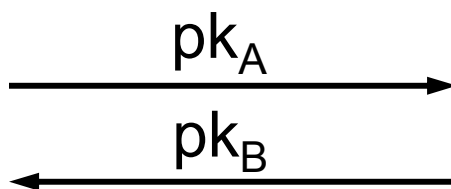
- Let \mathbf{G} be a group in which the **DDH** problem is **hard** and let g be a generator for \mathbf{G}

Alice

$$\begin{aligned} sk_A &\leftarrow \{0, \dots, |\mathbf{G}|-1\} \\ pk_A &\leftarrow g^{sk_A} \end{aligned}$$

Bob

$$\begin{aligned} sk_B &\leftarrow \{0, \dots, |\mathbf{G}|-1\} \\ pk_B &\leftarrow g^{sk_B} \end{aligned}$$



$$pk_B^{sk_A} = g^{sk_A sk_B} = pk_A^{sk_B}$$

Protocol does NOT provide authentication

Authenticated Key Exchange (AKE)

- Allow two parties to establish a common secret in an authenticated way
 - Parties should possess previously established authentication keys
- Intuitive goal: **implicit authentication**
 - The session key should only be known to the parties involved in the protocol
- Formally: **semantic security**
 - the session key should be *indistinguishable* from a random string

Further properties

- **Mutual authentication**

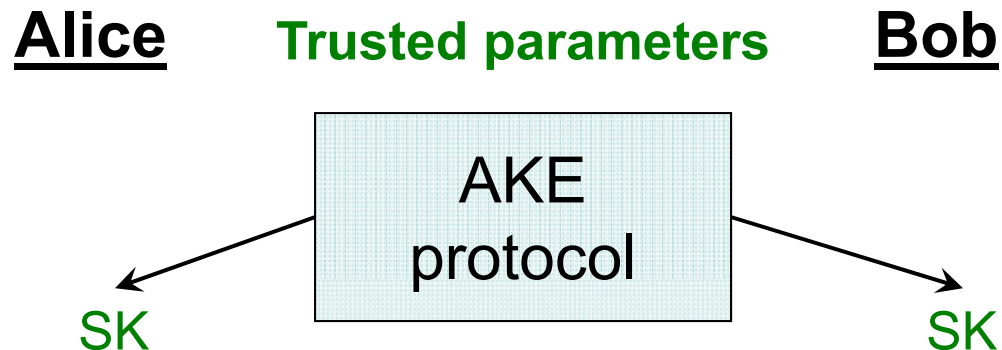
- Parties want to ensure that they *actually* share a secret with the other parties.

- **Forward secrecy**

- Previously established session keys will remain secure even after the corruption of the long-term secret.

AKE usage

- **AKE phase:**



- **Communication phase:**

- Alice and Bob use **SK** to secure their communication
- No assumption should be made about how **SK** is used

Authentication techniques

- **Asymmetric techniques**

- Assume the existence of a public-key infrastructure (PKI)
- Each party holds a pair of secret and public keys

- **Symmetric techniques**

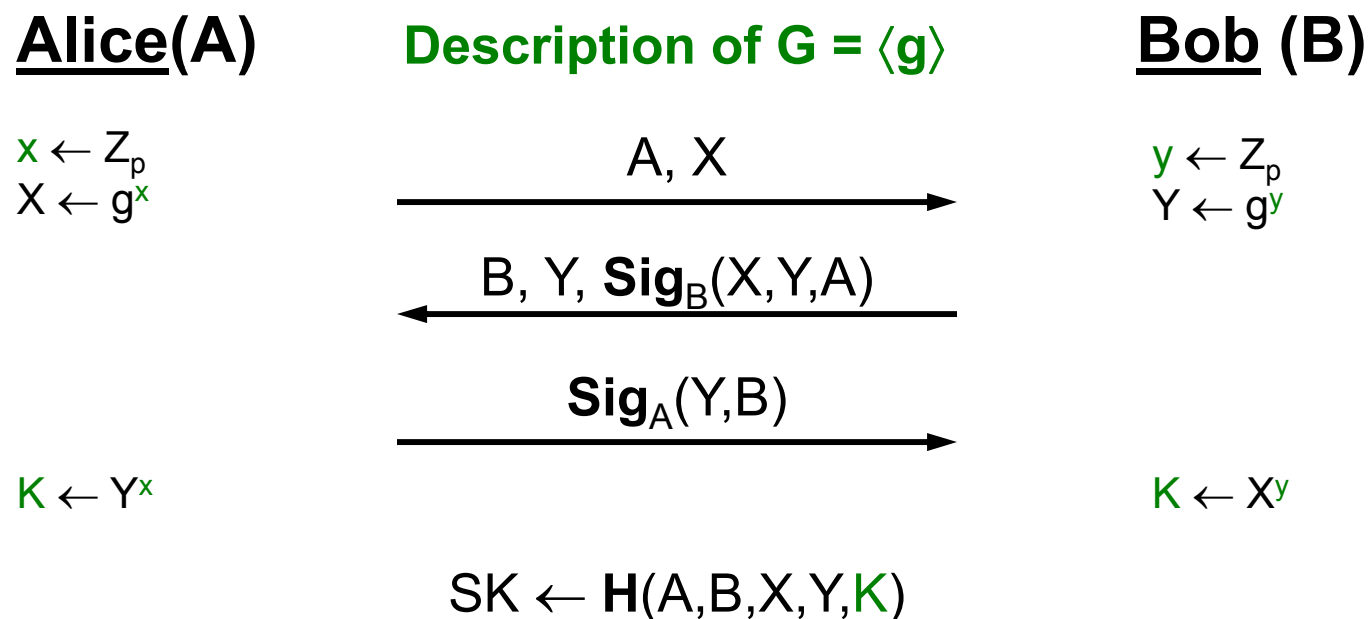
- Users share a random secret key
- **2-party** or **3-party** settings

- **Password-based techniques**

- Consider the case of weak secrets (e.g., a 4-digit PIN)
- Protocols are always subject to online guessing attacks

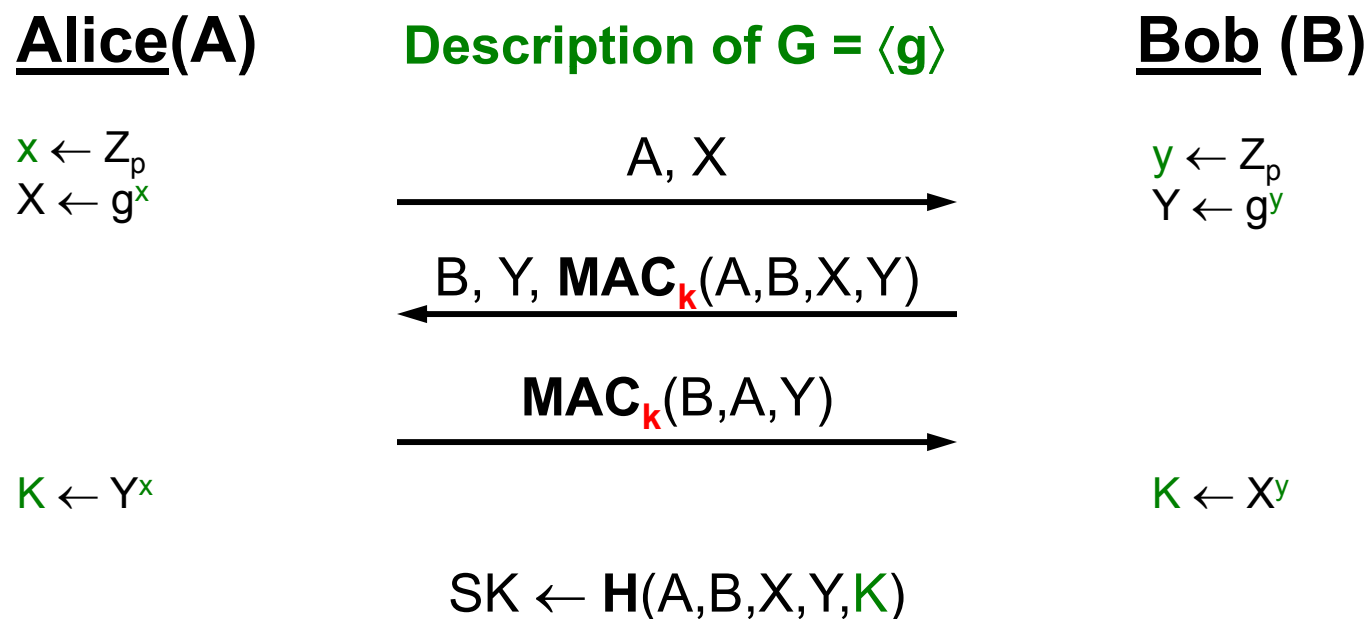
PKI-based techniques: Example

- The signed Diffie-Hellman protocol



Symmetric techniques: Example

- Alice and Bob share a MAC key **k**



Password-based authentication

- **Realistic**
 - Real-life applications usually rely on weak passwords
- **Convenient to use**
 - Users do not need to store the secret
- **Comes at a cost**
 - Protocols are always subject to online guessing attacks

Online dictionary attacks

- Let D represent the set of possible passwords (i.e., dictionary)
 - As passwords need be memorized by humans, $|D|$ is usually small
- **Online dictionary attack**
 - Choose a password from D
 - Interact with authentication server using the guessed password
 - Each online attempt can succeed with probability $1/|D|$
- **Counter measures against online attacks**
 - Limit the number of unsuccessful attempts
- **Goal of password-based authentication**
 - Restrict the adversary to online dictionary attacks only

Group AKE (GAKE)

- **Scenario**

- Similar to the 2-party case, except that ...
- Number of protocol participants is variable
- **Session key** is shared among all participants

- **Security goals**

- **Similar to the 2-party case:** Allow a group of users to establish a common session key with the help of their authentication keys

Outline

- Security model
- Unauthenticated key exchange schemes
- PKI-based authenticated key exchange
- ROM-based password-based AKE
- PAKE in the standard model
- Security in the UC framework
- Concluding remarks

Communication model

- Users can have many protocol instances running concurrently
 - Each user instance will be associated with an oracle
 - U^i represents the i -th instance of user U
- Communication may be controlled by the adversary
 - Adversary can create, modify, or forward messages
 - The transmission of messages is done via specific oracle queries
- Adversary is given oracle access to all user instances

Types of oracles

- **Execute(U_1^i, U_2^j)**
 - Models the eavesdropping on the protocol execution between user instances U_1^i and U_2^j
 - Since protocol is run over a public network, the transcripts are public
- **Reveal(U^i)**
 - Models the misuse of a session key by an user instance U^i
 - E.g., the secret session key may be exposed due to the use of a weak encryption scheme

Types of oracles

- **Send(U^i)**

- Models active attacks
- Allows the adversary to send message to user instance U^i in place of another user instance
- Allows the adversary to intercept, forward and/or modify messages

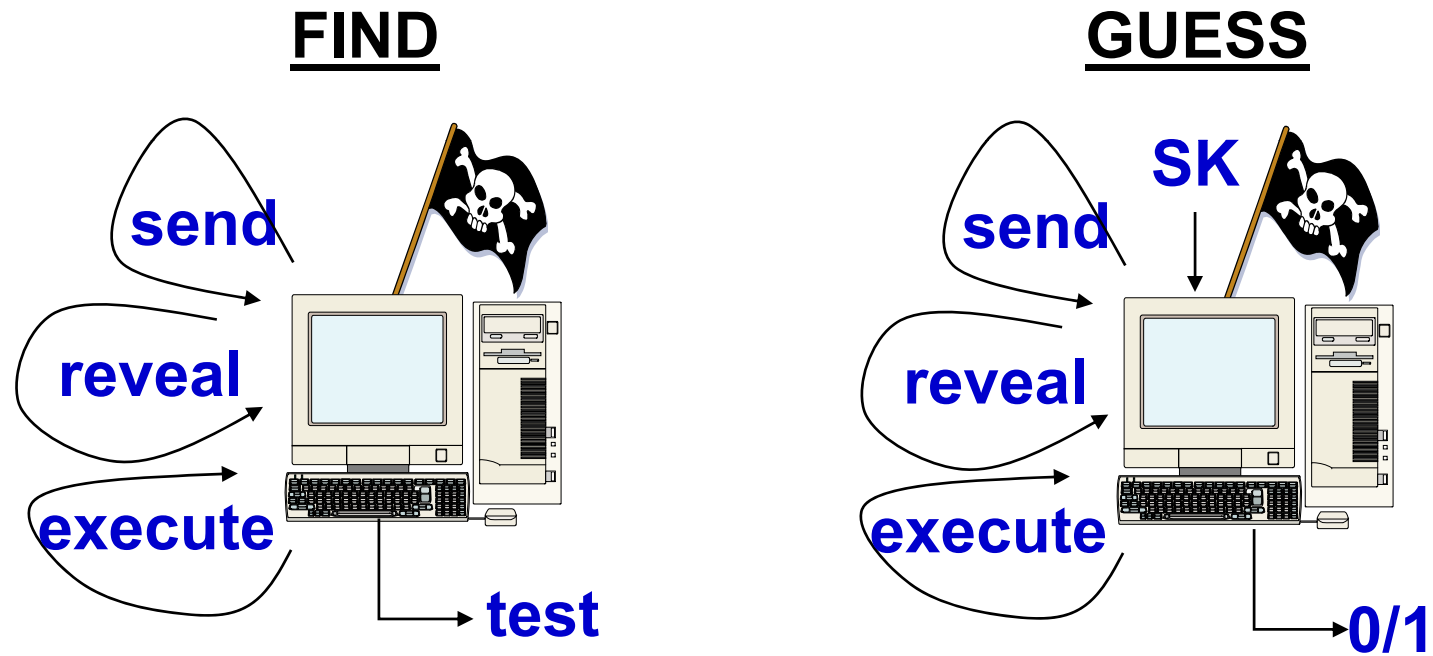
- **Test(U^i)**

- Models the semantic security of the session key held by U^i
- The output is either the real session key held by U^i ($b=0$) or a random key ($b=1$)

Passive/Active Adversaries

- **Passive adversary:**
History built using
 - Execute queries → transcripts
 - Reveal queries → exposure of session keys
- **Active adversary:**
Entire control of the network via send queries

The security experiment



Goal: Guess whether **SK** is the actual session key

Auxiliary definitions

- **Partnering**

- Two instances are said to be **partners** if they share the same session identification (**sid**)
- The probability of two sessions having the same **sid** should be negligible

- **Freshness**

- Avoids cases in which the adversary can trivially break the semantic security
- An instance U^i is said to be **fresh** if U^i has **accepted** and **no reveal query** has been asked to U^i or to its partner

Security definition

- Let **SUCC** denote the success probability of the adversary in guessing the hidden bit used in the test query

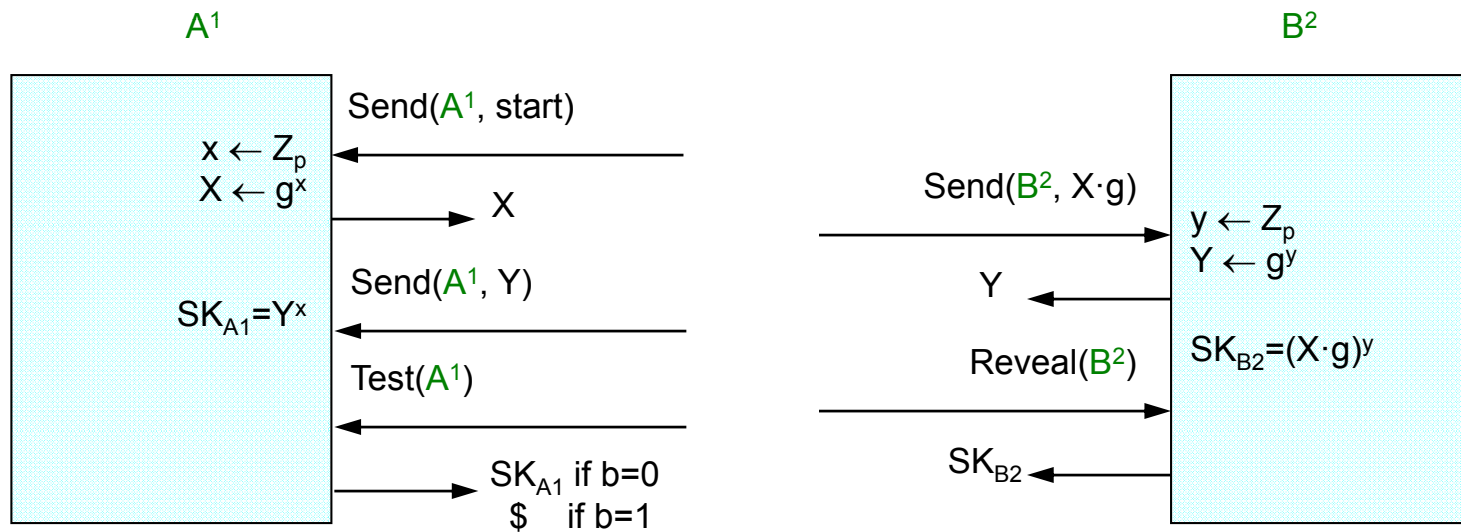
$$\text{Adv}_P^{\text{ake}}(t, q_{\text{reveal}}, q_{\text{send}}, q_{\text{exe}}) = \max_A \{2 \Pr[\text{SUCC}] - 1\}$$

- q_{reveal} , q_{send} , and q_{exe} are the maximum number of queries made by A to the reveal, send, and execute oracles.

Secure PAKE: $\text{Adv}^{\text{AKE}}_P() \approx q_{\text{send}}/|\text{Dictionary}| + \text{negl}(k)$

Exercising the model

- A simple man-in-the-middle attack against the (unauthenticated) Diffie-Hellman protocol



If $SK_{A^1} = SK_{B^2} \cdot Y$, then output 0 else output 1

Outline

- ✓ Security model
- Unauthenticated key exchange protocols
- PKI-based authenticated key exchange
- ROM-based Password-based AKE
- PAKE in the standard model
- Security in the UC framework
- Concluding remarks

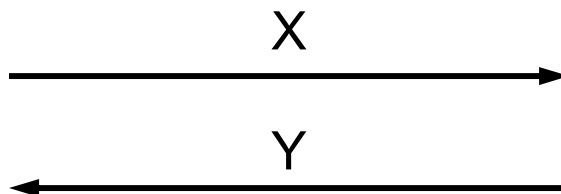
The Diffie-Hellman protocol

Alice

$$x \leftarrow \mathbb{Z}_p$$
$$X \leftarrow g^x$$

$$K \leftarrow Y^x$$

Description of $G = \langle g \rangle$



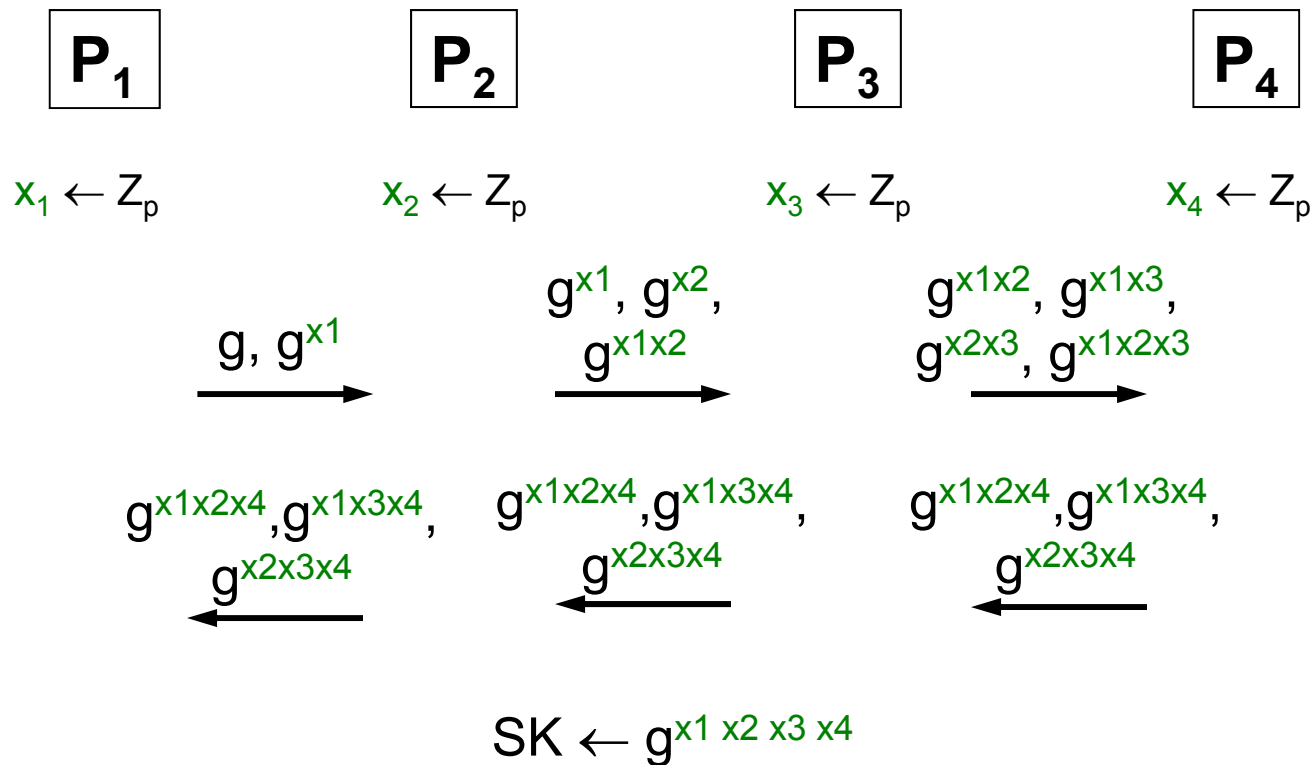
Bob

$$y \leftarrow \mathbb{Z}_p$$
$$Y \leftarrow g^y$$

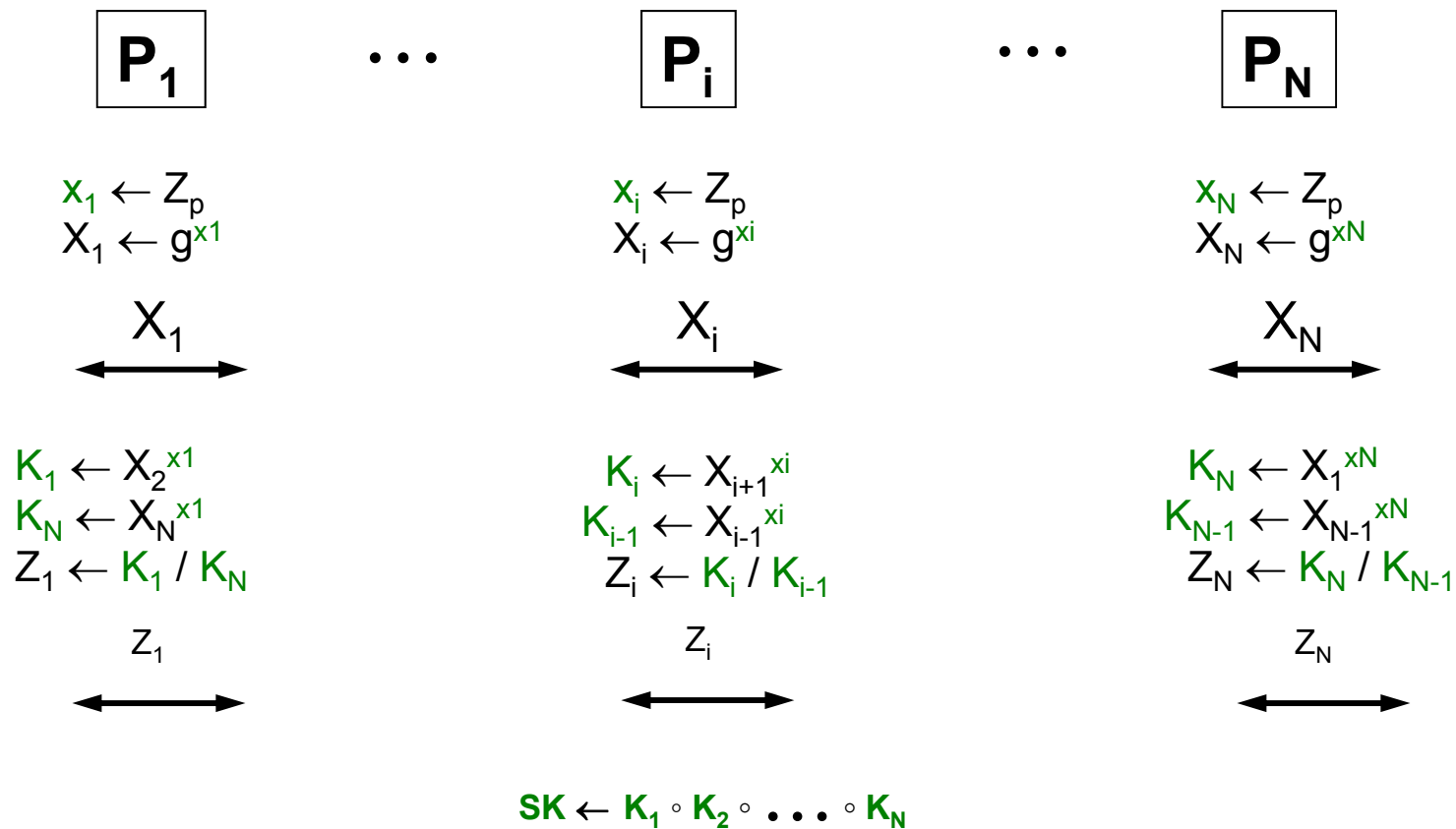
$$K \leftarrow X^y$$

$$K \leftarrow g^{xy}$$

Steiner-Tsudik-Waidner group key exchange [STW96]

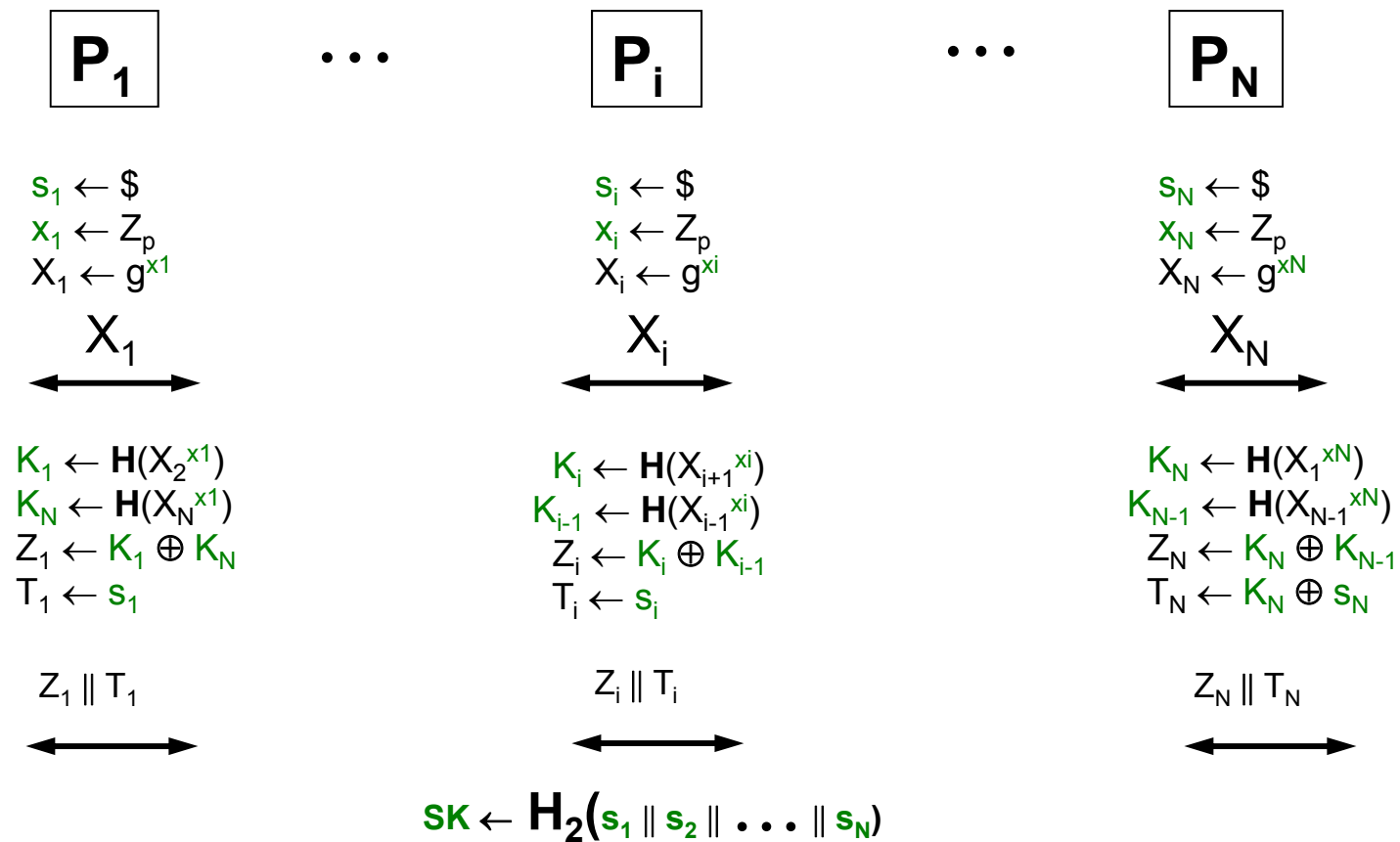


Burmester-Desmedt group key exchange [BD94]

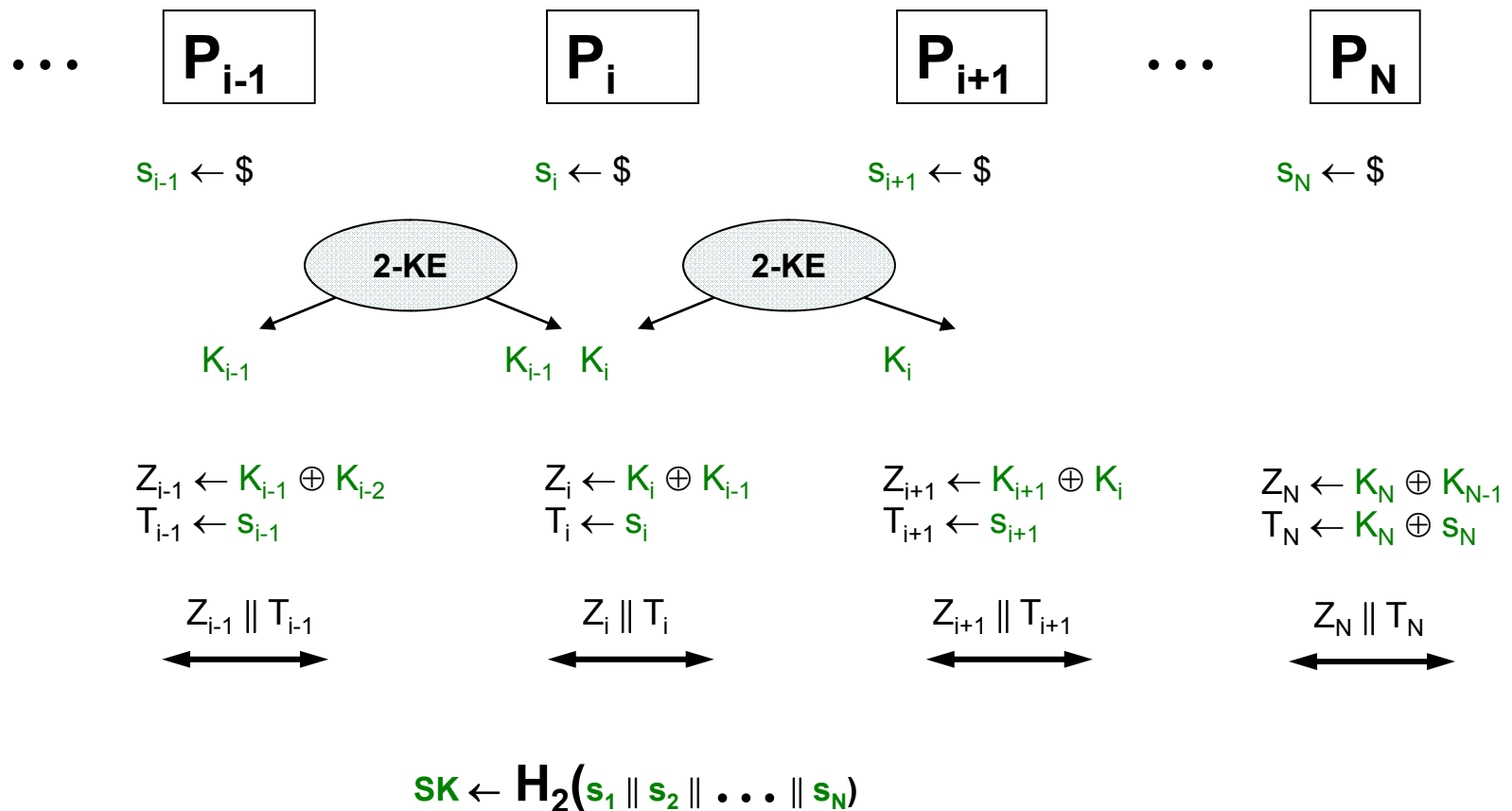


Kim-Lee-Lee GKE scheme

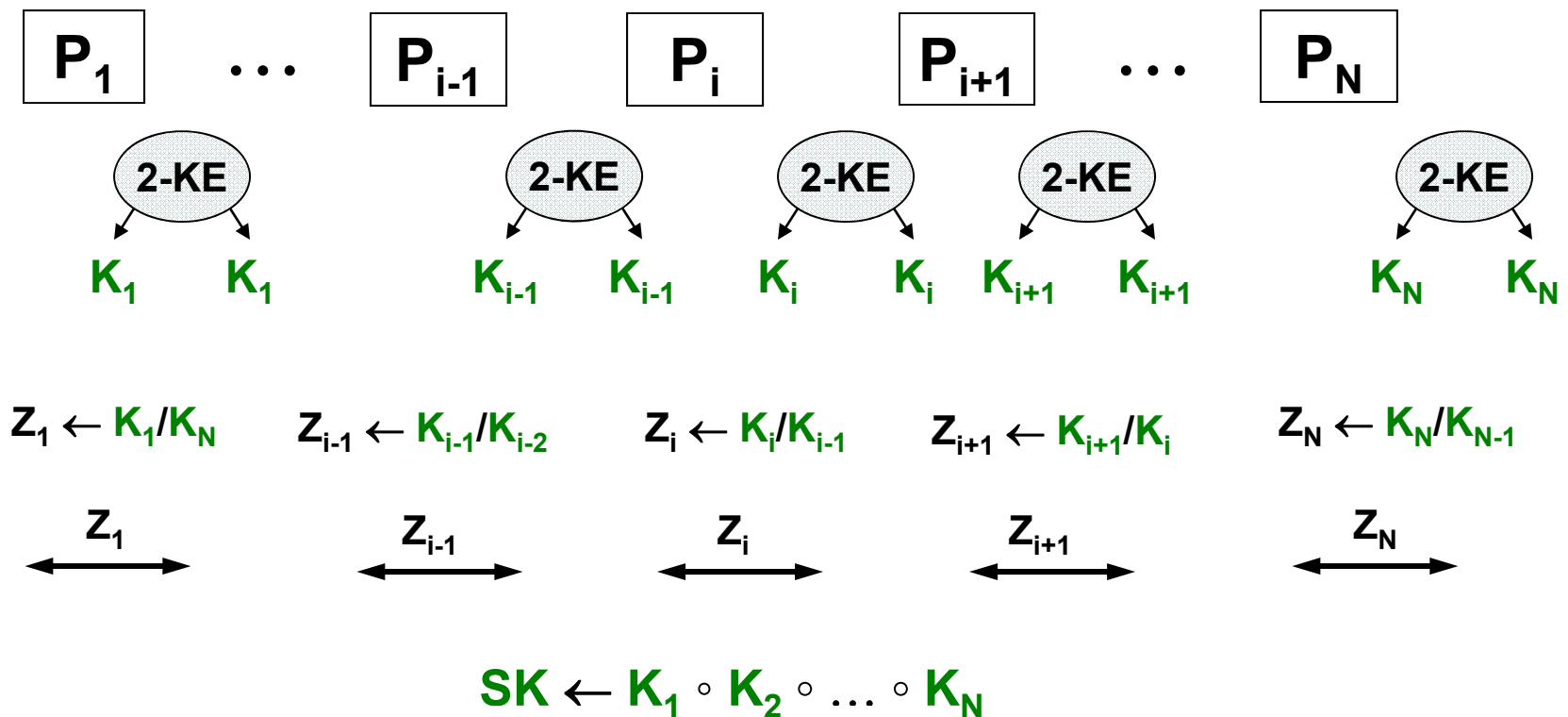
[KLL04]



A generic version of the Kim-Lee-Lee protocol



A generic version of the Burmester-Desmedt protocol



Security of Generic Burmester-Desmedt

If

- the underlying 2-party key exchange 2-KE is secure against passive attacks

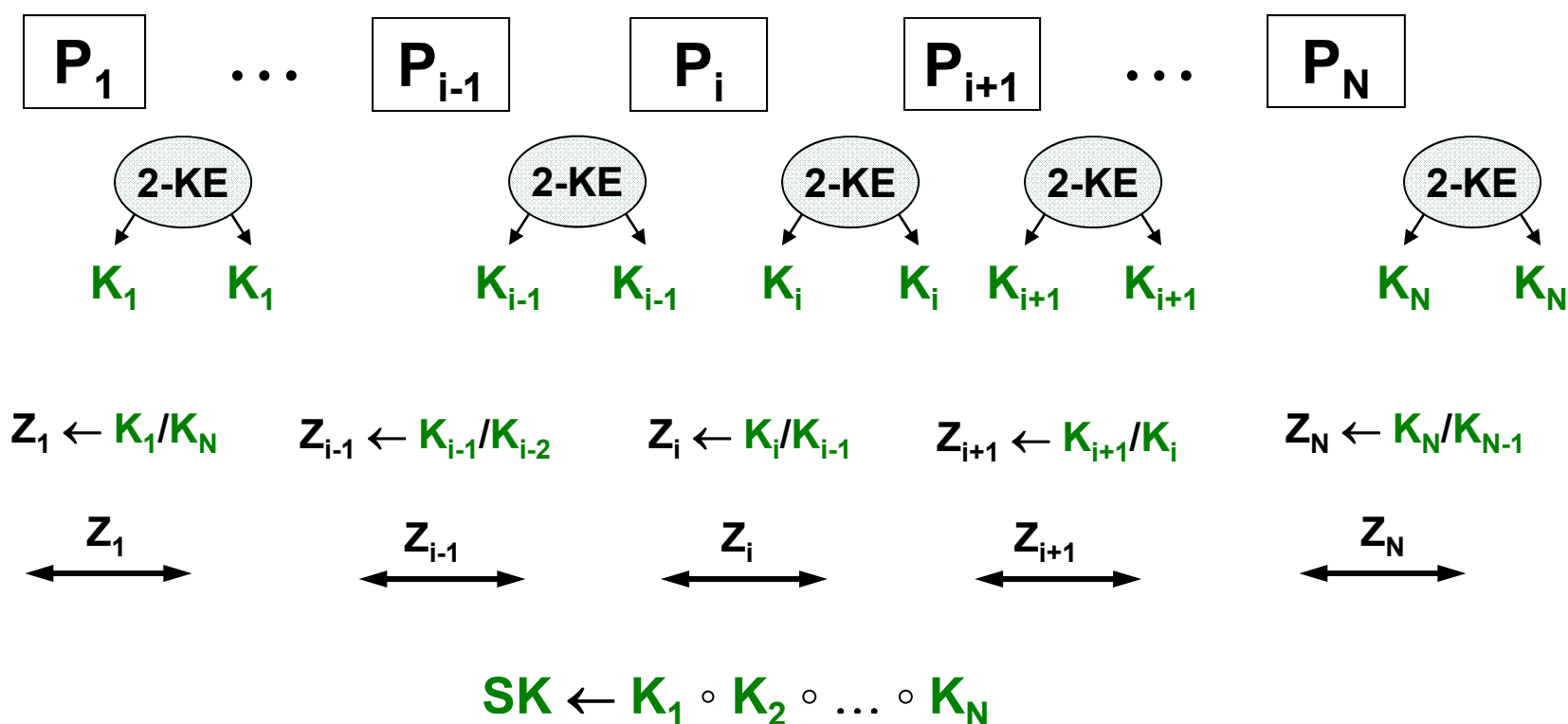
Then

- the generic Burmester-Desmedt protocol is **secure against** passive attacks.

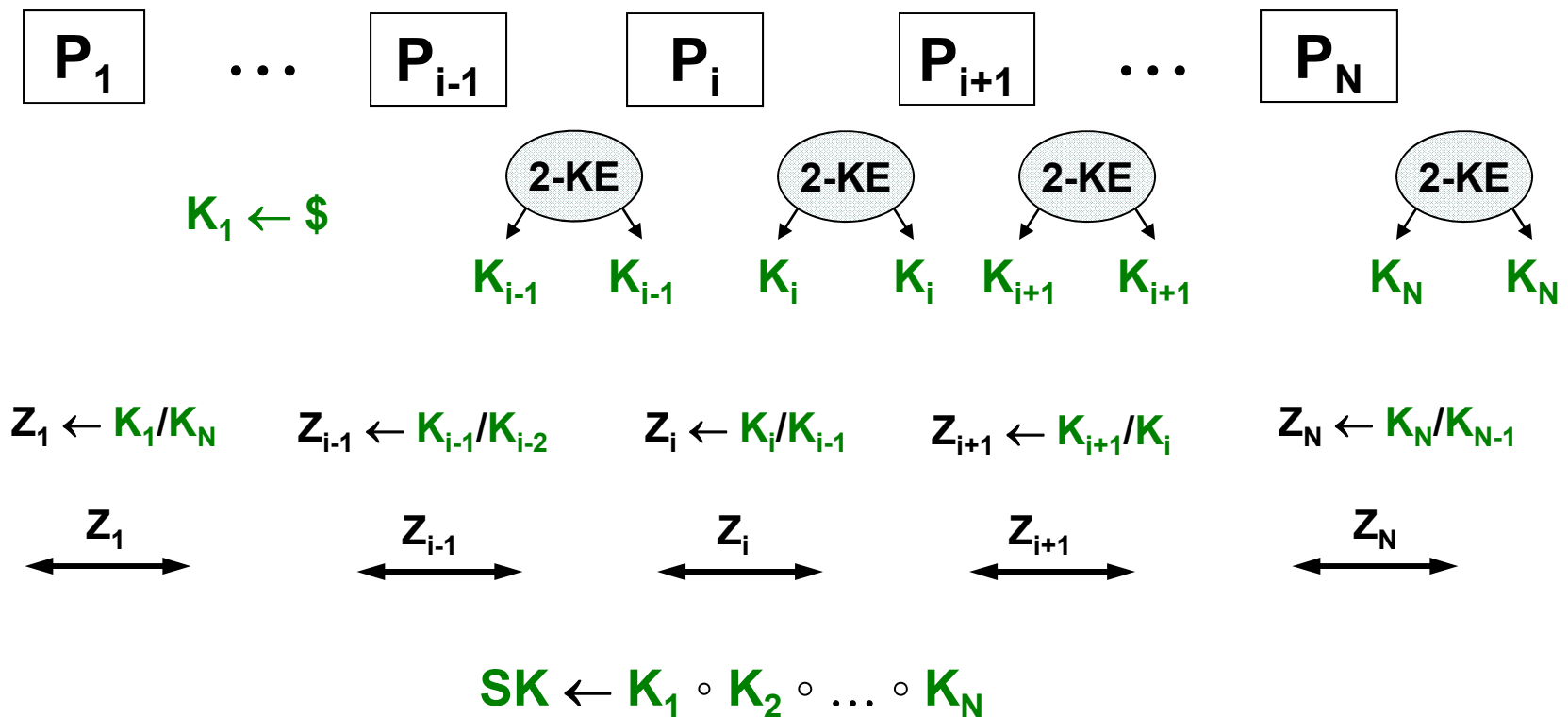
Proof by games

- Define sequence of games $\mathbf{G}_0, \dots, \mathbf{G}_N, \mathbf{G}_{N+1}$
- \mathbf{G}_0 : The real game
 - $\text{Adv}_0 = \text{Adv}_{\text{BD}}(A)$
- \mathbf{G}_1 : Replace secret key \mathbf{SK}_1 by random key
 - $|\text{Adv}_1 - \text{Adv}_0| \leq \text{Adv}_{\text{KE}}(A) = \text{negl}(k)$
- ...
- \mathbf{G}_N : Replace secret key \mathbf{SK}_N by random key
 - $|\text{Adv}_N - \text{Adv}_{N-1}| \leq \text{Adv}_{\text{KE}}(A) = \text{negl}(k)$
- \mathbf{G}_{N+1} : Replace session key \mathbf{SK} by random key
 - $\text{Adv}_N = \text{Adv}_{N-1}$ (N variables, N-1 equations)
 - $\text{Adv}_{N-1} = 0$

Game 0: the real protocol



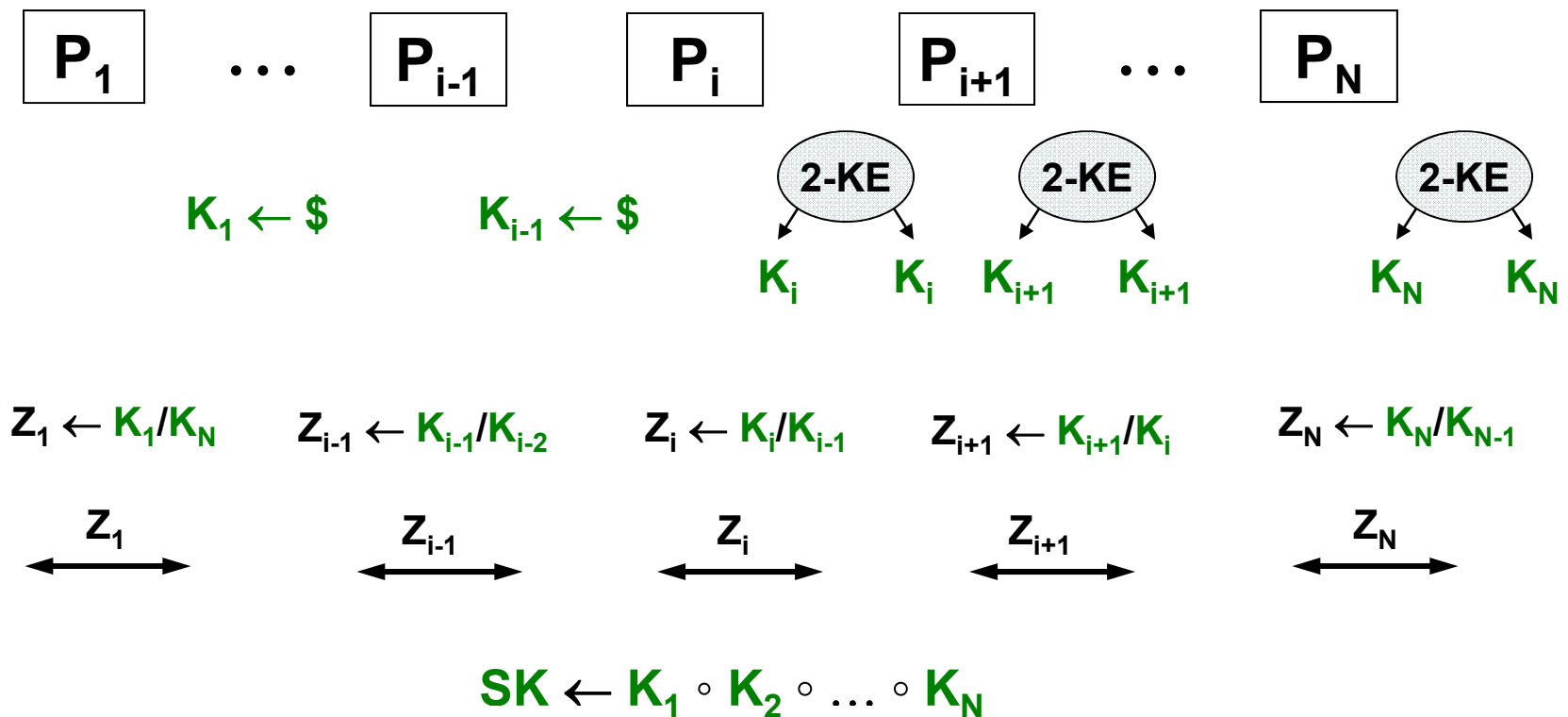
Game 1: $SK_1 = \text{random}$



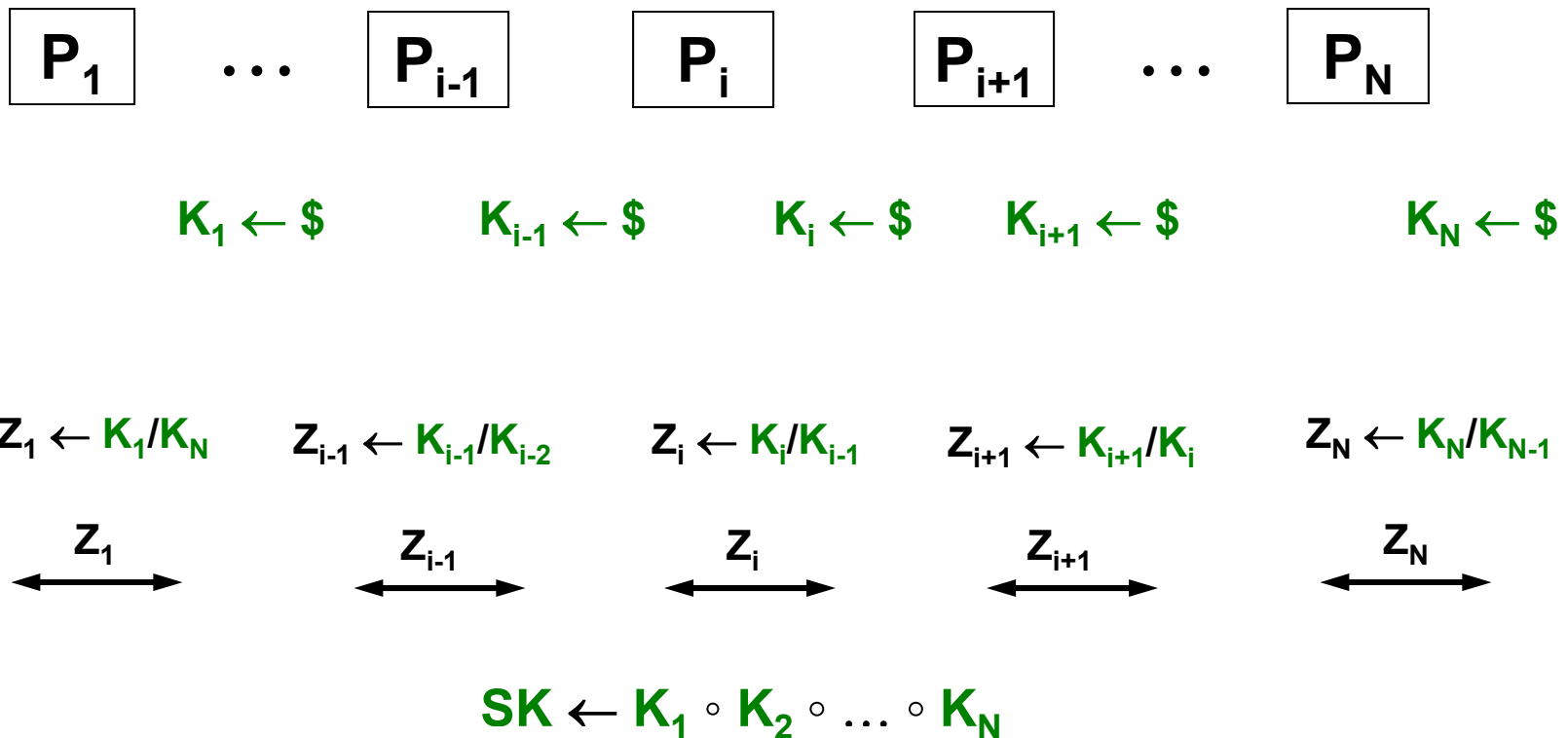
Reduction

- Guess challenge session \Rightarrow $1/q$ of being right
- **Execute oracle simulation**
 - Simulate 2-KE protocol between U_1 and U_2 using **2-KE execute oracle**
 - Simulate 2-KE protocols between other players as in the real protocol
 - If challenge session, then compute K_1 using **2-KE test oracle**. Else, compute K_1 using **2-KE reveal oracle**
 - Compute remaining values as in the real protocol
- **Reveal oracle simulation**
 - Simulate as in the real protocol
- **Test oracle simulation**
 - Simulate as in the real protocol

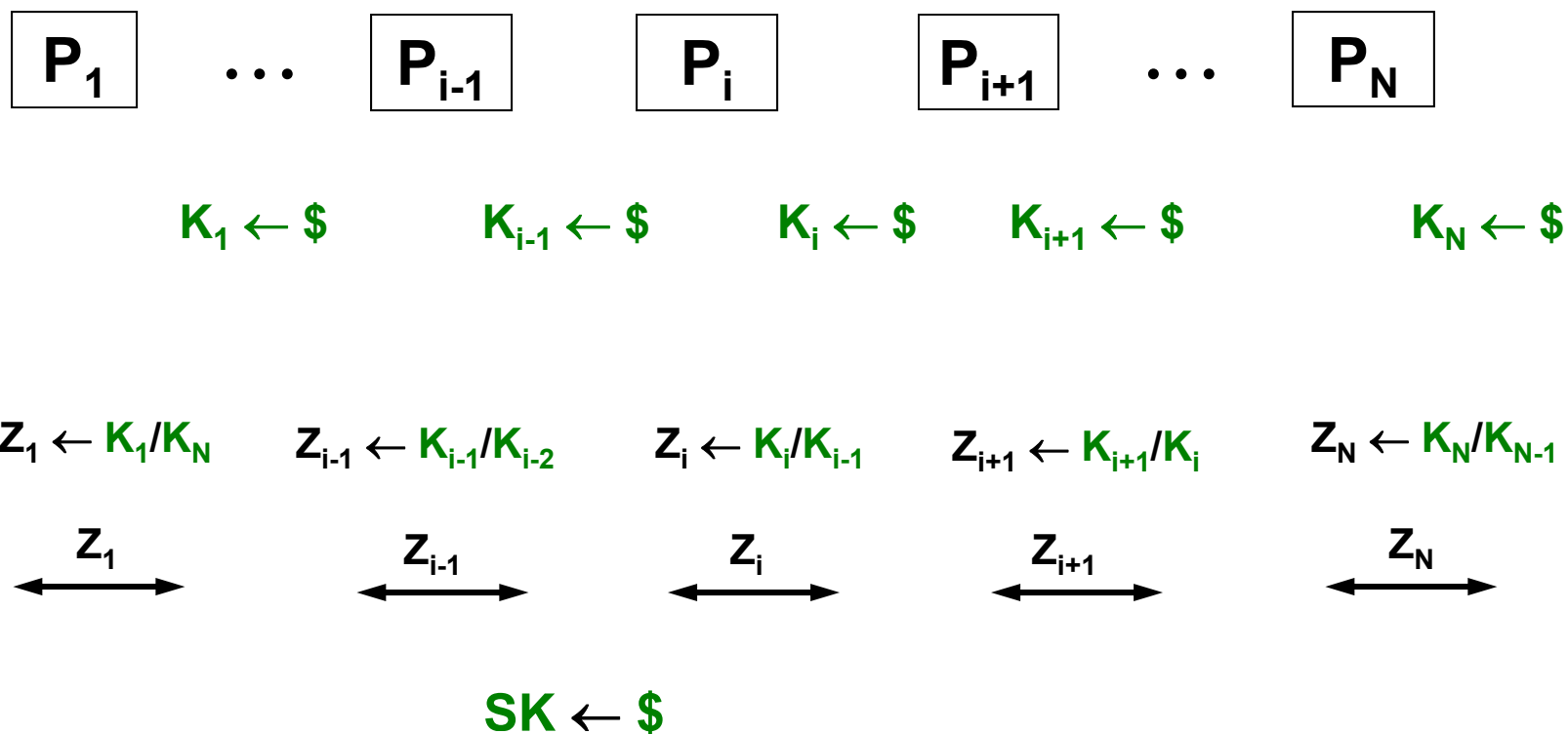
Game i: $SK_{i-1} = \text{random}$



Game N: $SK_N = \text{random}$



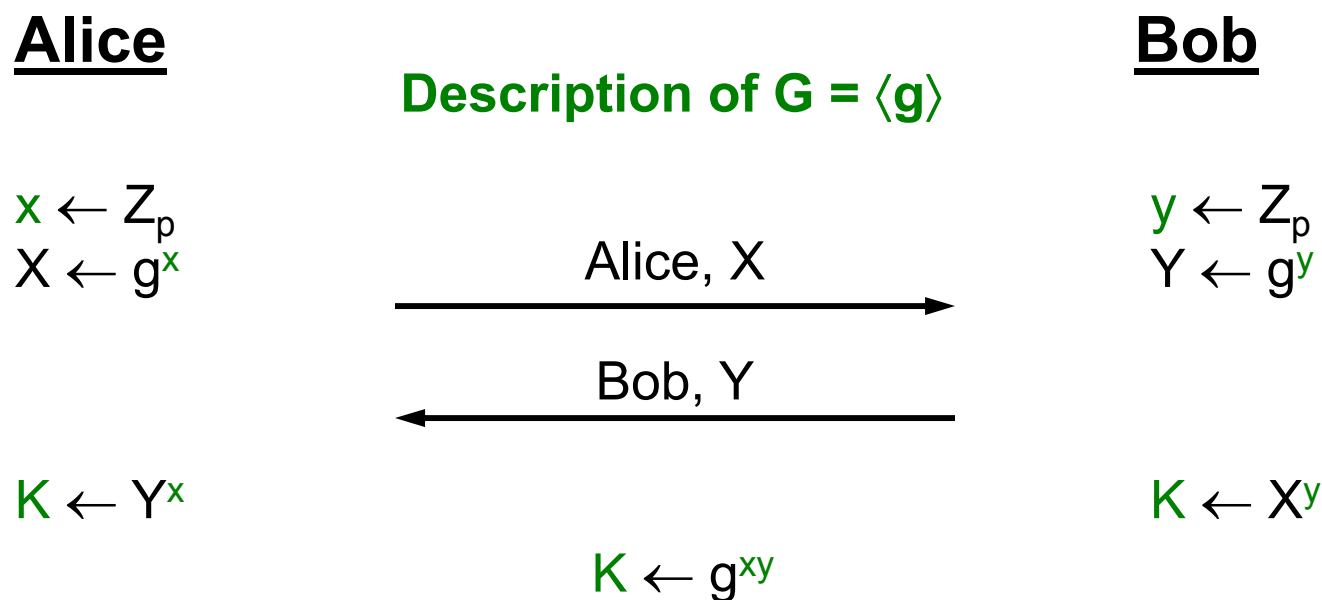
Game N+1: SK = random



Outline

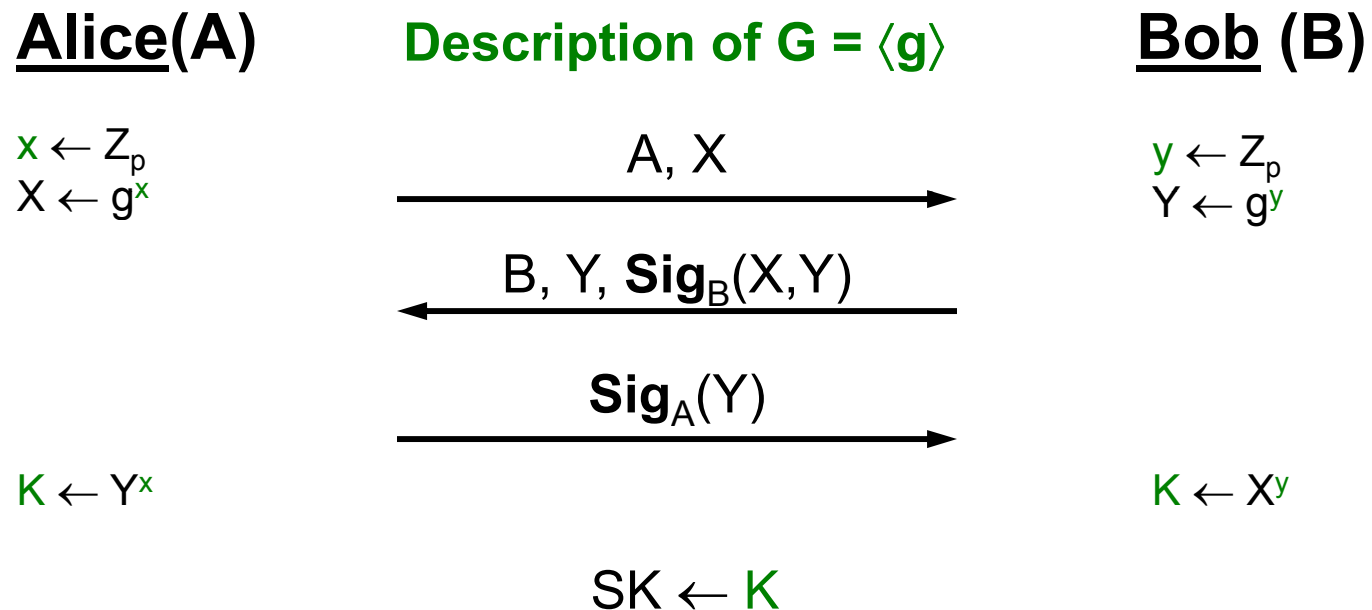
- ✓ Security model
- ✓ Unauthenticated key exchange protocols
- PKI-based authenticated key exchange
- ROM-based password-based AKE
- PAKE in the standard model
- Security in the UC framework
- Concluding remarks

The Diffie-Hellman protocol



- How can we add authentication to DH?
- Would signing each message be enough?

BADH: Basic authenticated Diffie-Hellman (Example by Krawczyk)



- Output:
 - Alice: Shares $K=g^{xy}$ with Bob
 - Bob: Shares $K=g^{xy}$ with Alice

Problem with BADH: Identity misbinding attack

Alice(A)

$$x \leftarrow Z_p$$

$$X \leftarrow g^x$$

$\xrightarrow{A, X}$

$\xleftarrow{B, Y, \text{Sig}_B(X, Y)}$

$\xrightarrow{\text{Sig}_A(Y)}$

$$K \leftarrow Y^x$$

$$\text{SK} \leftarrow K$$

Eve(E)

$$E, X$$

$\xrightarrow{E, X}$

$\xleftarrow{B, Y, \text{Sig}_B(X, Y)}$

$\xrightarrow{\text{Sig}_E(Y)}$

$$\text{SK} \leftarrow K$$

Bob (B)

$$y \leftarrow Z_p$$

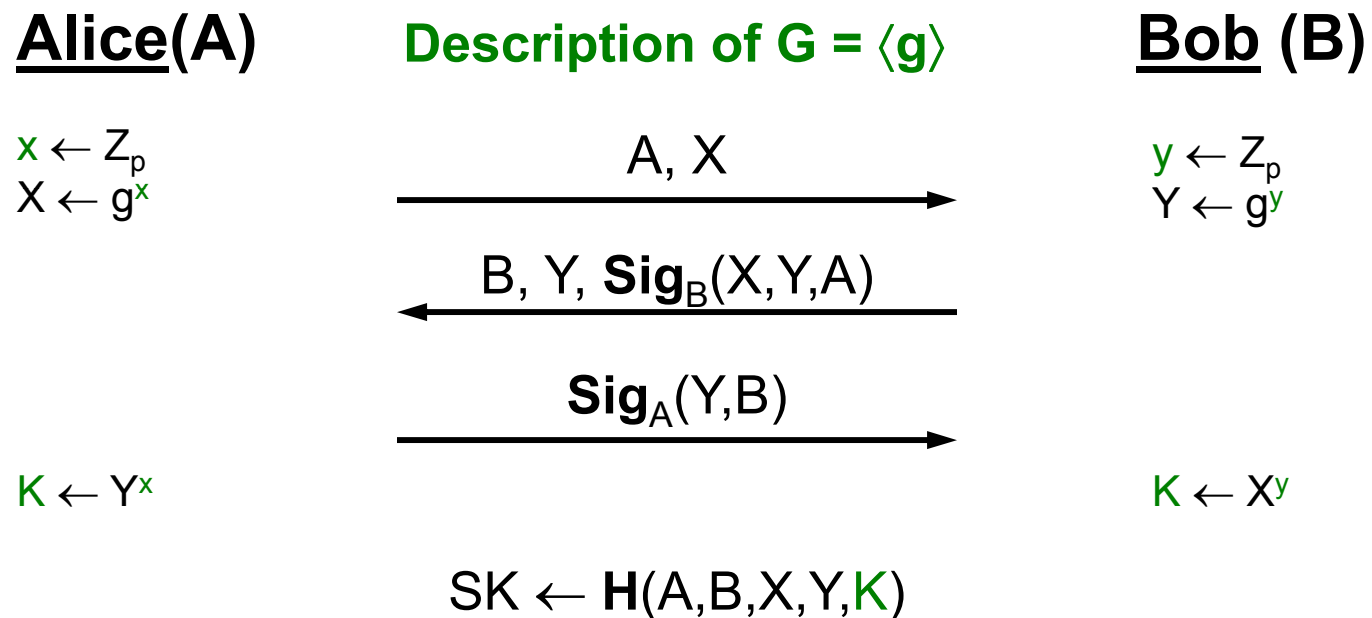
$$Y \leftarrow g^y$$

$$K \leftarrow X^y$$

- **Output:**

- Alice: Shares $K=g^{xy}$ with Bob
- Bob: **Shares $K=g^{xy}$ with Eve**

Signed Diffie-Hellman protocol (ISO-9796)



- Identity misbinding attack is avoided by including the identity of the peer in the signature

Adding authentication to group key exchange protocols

- **Lessons from the 2-party setting**
 - Adding authentication can be tricky and potentially yield insecure protocols
 - Signing each message may be enough, but how can we be sure that the resulting protocol is secure?
- **Generic compilers**
 - **Resulting GKE** should be secure against active attacks if **underlying GKE** is secure against passive attacks

The Katz-Yung compiler

- Given GKE \mathbf{P} among $G_{ID} = \{U_1, \dots, U_N\}$, generate GKE \mathbf{P}_{active} as follows:
 - **Step 1: Add a round of nonces**
 - Each U_i chooses r_i in $\{0,1\}^k$ and broadcasts $U_i \parallel 0 \parallel r_i$
 - **Step 2: Number each message and sign it**
 - Whenever U_i broadcasts message $U_i \parallel j \parallel M$ in \mathbf{P} , attach signature σ for $U_i \parallel j \parallel M \parallel r_1 \parallel \dots \parallel r_N$
 - Whenever U_i receives message $V \parallel j \parallel M \parallel \sigma$, check that V is in G_{ID} and that signature is correct

Outline

- ✓ Security model
- ✓ Unauthenticated key exchange protocols
- ✓ PKI-based authenticated key exchange
- ROM-based password-based AKE
- PAKE in the standard model
- Security in the UC framework
- Concluding remarks

Background: Ideal models

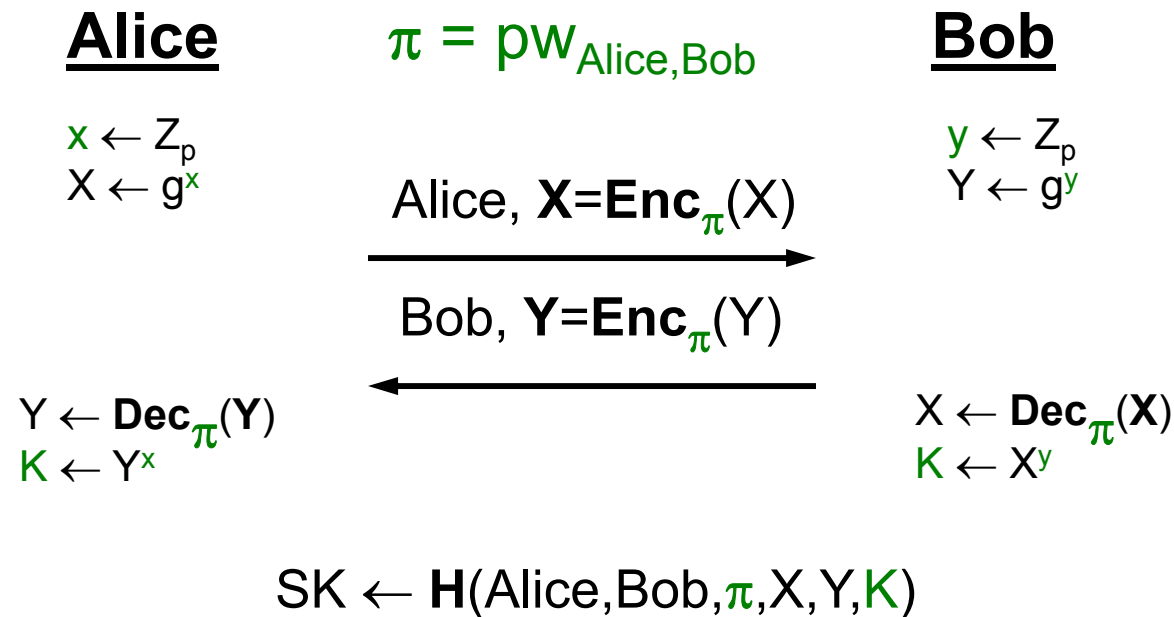
- Random oracle model [BellareRogaway93]
 - Perhaps the most used ideal model in cryptography
 - The hash function is modeled as a perfectly random function
- Random permutation model
 - Similar to the random-oracle model, but with a permutation instead of a function
- Ideal cipher model
 - An extension of the random-permutation model
 - A block cipher is seen as a family of truly random and independent permutations (for each key)
- Standard model
 - None of the above

Brief history of PAKE schemes

- **[BelMer92]: Encrypted Key Exchange (EKE)**
 - Seminal work, no proofs
- **[BelPoiRog00,BoyMacPat00]**
 - Formal security models
 - Protocols in the ideal-cipher and random-oracle models
- **[GoLin01]**
 - Non-concurrent protocol in the standard model
- **[KatOstYun01,GenLin03,CHKLM05]**
 - Efficient protocols in the CRS model
- **[BR00,BCP03/04,CatPoiPor04,MacKenzie02,AbPo05]**
 - Efficient EKE and OKE protocols in the RO model

Encrypted Key Exchange [Bellare & Merritt, 1992]

- Flows are encrypted with the password



EKE instantiations

- **[BPR00,BCP03]**
 - **Enc** = Ideal cipher
 - **H** = Random oracle
- **[MacKenzie02,BCP04]**
 - **Enc** = Random oracle
 - **H** = Random oracle
- **[AbPo05]**
 - **Enc**_{pw}(X) = X * h^{pw}
 - **H** = Random oracle

Simple PAKE [AbPo05]

Alice

$$x \leftarrow Z_p$$
$$X \leftarrow g^x$$

$$Y \leftarrow Y / B^\pi$$
$$K \leftarrow Y^x$$

$$\pi = \rho W_{\text{Alice,Bob}}$$

$$\text{Alice, } X = X * A^\pi$$

$$\text{Bob, } Y = Y * B^\pi$$

Bob

$$y \leftarrow Z_p$$
$$Y \leftarrow g^y$$

$$X \leftarrow X / A^\pi$$
$$K \leftarrow X^y$$

$$SK \leftarrow H(\text{Alice,Bob}, \pi, X, Y, K)$$

Security of SPAKE

If

- the **CDH** is hard in the underlying group and
- **H** is a random oracle

Then

- SPAKE is a **secure against** active attacks
- $\text{Adv}(A) \leq O(q_{\text{send}}/|\mathbf{Dictionary}|) + \text{negl}(k)$

Proof by games

- Define sequence of games $\mathbf{G}_0, \dots, \mathbf{G}_4$
- \mathbf{G}_0 : The real game
 - $\text{Adv}_0 = \text{Adv}_{\text{SPAKE}}(A)$
- \mathbf{G}_1 : Simulate **execute**, **reveal**, **send** oracles as in the real game
 - $\text{Adv}_1 = \text{Adv}_0$
- \mathbf{G}_2 : abort whenever there is a collision in the transcript $(A, \mathbf{X}), (B, \mathbf{Y})$
 - $|\text{Adv}_2 - \text{Adv}_1| \leq \text{negl}(k)$
- \mathbf{G}_3 : $\text{SK} = \mathbf{H}'(\text{Alice}, \text{Bob}, \mathbf{X}, \mathbf{Y})$ and $\text{pw} = 0$ in **execute** oracle simulation
 - $|\text{Adv}_3 - \text{Adv}_2| \leq \text{negl}(k)$ if **CDH** is hard
- \mathbf{G}_4 : $\text{SK} = \mathbf{H}'(\text{Alice}, \text{Bob}, \mathbf{X}, \mathbf{Y})$ and $\text{pw} = 0$ in **send** oracle simulation
 - $|\text{Adv}_4 - \text{Adv}_3| \leq O(q_{\text{send}} / |\text{Dictionary}|) + \text{negl}(k)$ if **CDH** is hard
 - $\text{Adv}_{N-1} = 0$

Brief history of GAKE schemes

- **[BurDes94, BurDes05]:**
 - Constant-round group Diffie-Hellman key exchange
 - Passive attacks, security based on CDH
- **[KatzYung03]**
 - Proof of security for BD protocol based on DDH
 - Generic compiler from GKE to GAKE [using signatures](#)
- **[KimLeeLee04]**
 - A variant of the BD protocol using random oracles and XOR operations
- **[Joux00]**
 - A One Round Protocol for Tripartite Diffie-Hellman
- **[LiPieprzyk99,BreCat04]**
 - Conference key agreement from secret sharing
- **[BoydNieto03, JeongKatzLee04, ...]**
 - Round-Optimal contributory key agreement

Previous work on GPAKE

- **[BreChePoi02, BreChePoi05]:**
 - Group Diffie-Hellman password-based key exchange
 - Linear number of rounds
- **[LeeHwangLee04]**
 - Based on the Kim-Lee-Lee GAKE protocol
 - Proven secure in the random-oracle model
 - **Broken in [ABCP06]**
- **[DuttaBarua06]**
 - Based on the Kim-Lee-Lee GAKE protocol
 - Proven secure in the random-oracle and ideal-cipher models
 - **Broken in [ABCP06]**
- **[ABCP06], [TangChoo06]**
 - Based on the Burmester-Desmedt protocol
 - Proven secure in the ideal-cipher and random-oracle models

More recent work on GPAKE

- **[KwonJeongLee06]**
 - Simplification of [ABCP06] protocol
 - Proven secure in the “standard model”
 - **Apparently insecure**
- **[AbdallaPointcheval06]**
 - Based on the Gennaro-Lindell PAKE protocol
 - Proven secure in the standard model
- **[BohliGonzalezSteinwandt06]**
 - Proven secure in the standard model
 - Similar to [AbdallaPointcheval06], but more efficient
- **[ABGS07+Nem10]**
 - Generic compiler from 2-party to group
 - Proven secure in the standard model

Adding password authentication to the BD protocol

- **EKE approach**

- Encrypt all flows using the password **pw**
- $X_i = E_{pw}(X_i)$, $Z_i = E_{pw}(Z_i)$

- **Problem**

- In the BD protocol, $Z_1 \circ Z_2 \circ \dots \circ Z_N = 1$
- Dictionary attack
 - Guess password **pw**
 - Compute $Z_i = D_{pw}(Z_i)$ for $i=1, \dots, N$
 - Check if $Z_1 \circ Z_2 \circ \dots \circ Z_N = 1$

Dutta-Barua GPAKE [DB06]

P_1

$$\begin{aligned} s_1 &\leftarrow \$ \\ x_1 &\leftarrow Z_p \\ X_1 &\leftarrow g^{x_1} \end{aligned}$$

$$\longleftrightarrow E_{pw}(X_1)$$

$$\begin{aligned} K_1 &\leftarrow H(X_2^{x_1}) \\ K_N &\leftarrow H(X_N^{x_1}) \\ Z_1 &\leftarrow K_1 \oplus K_N \\ T_1 &\leftarrow s_1 \end{aligned}$$

$$\longleftrightarrow E'_{pw}(Z_1 || T_1)$$

P_i

$$\begin{aligned} s_i &\leftarrow \$ \\ x_i &\leftarrow Z_p \\ X_i &\leftarrow g^{x_i} \end{aligned}$$

$$\longleftrightarrow E_{pw}(X_i)$$

$$\begin{aligned} K_i &\leftarrow H(X_{i+1}^{x_i}) \\ K_{i-1} &\leftarrow H(X_{i-1}^{x_i}) \\ Z_i &\leftarrow K_i \oplus K_{i-1} \\ T_i &\leftarrow s_i \end{aligned}$$

$$\longleftrightarrow E'_{pw}(Z_i || T_i)$$

P_N

$$\begin{aligned} s_N &\leftarrow \$ \\ x_N &\leftarrow Z_p \\ X_N &\leftarrow g^{x_N} \end{aligned}$$

$$\longleftrightarrow E_{pw}(X_N)$$

$$\begin{aligned} K_N &\leftarrow H(X_1^{x_N}) \\ K_{N-1} &\leftarrow H(X_{N-1}^{x_N}) \\ Z_N &\leftarrow K_N \oplus K_{N-1} \\ T_N &\leftarrow K_N \oplus s_N \end{aligned}$$

$$\longleftrightarrow E''_{pw}(T_N)$$

$$SK \leftarrow H_2(s_1 || s_2 || \dots || s_N)$$

An attack against the Dutta-Barua GPAKE protocol

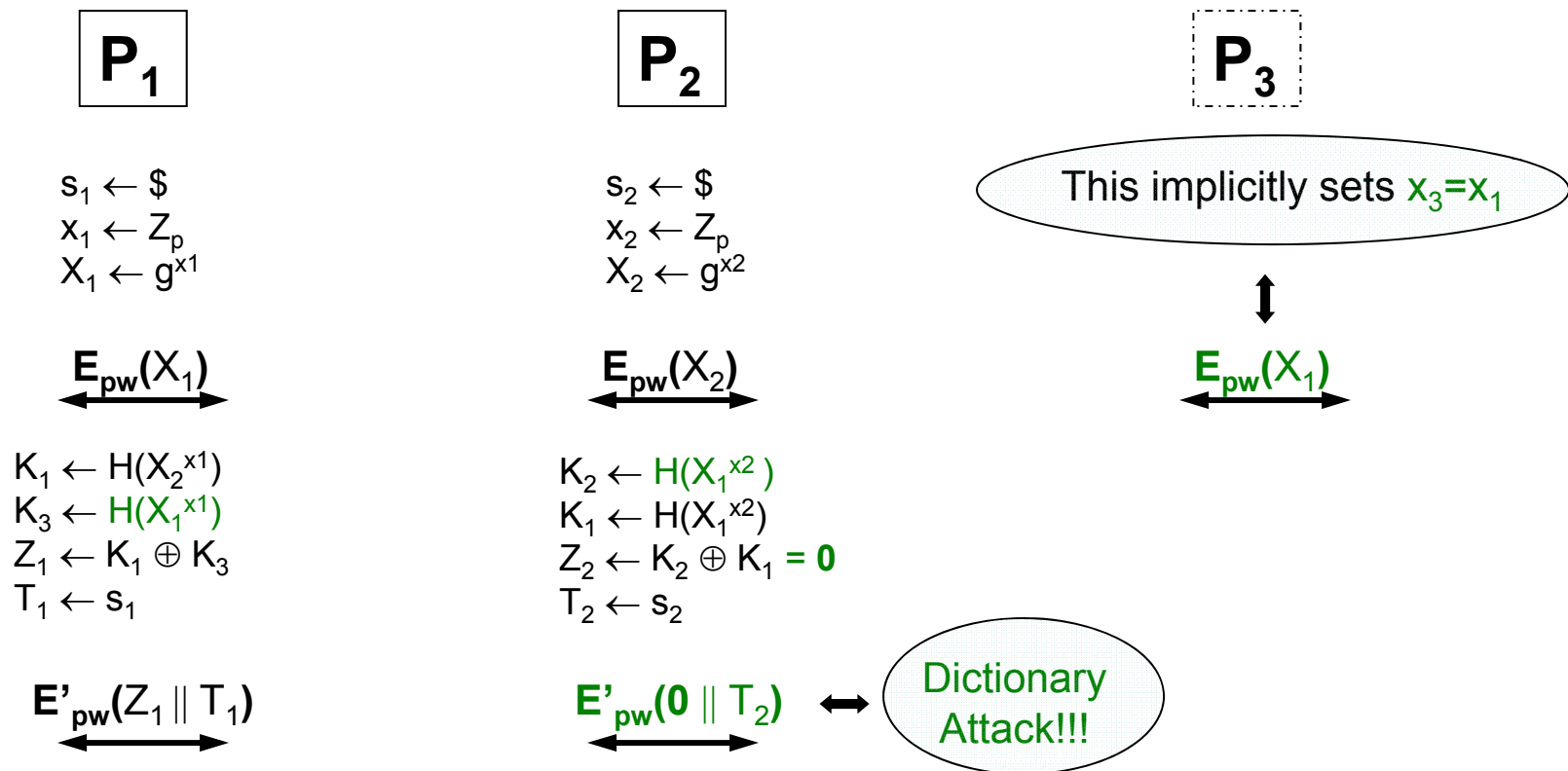
- **Problem**

- All flows are encrypted under the same key

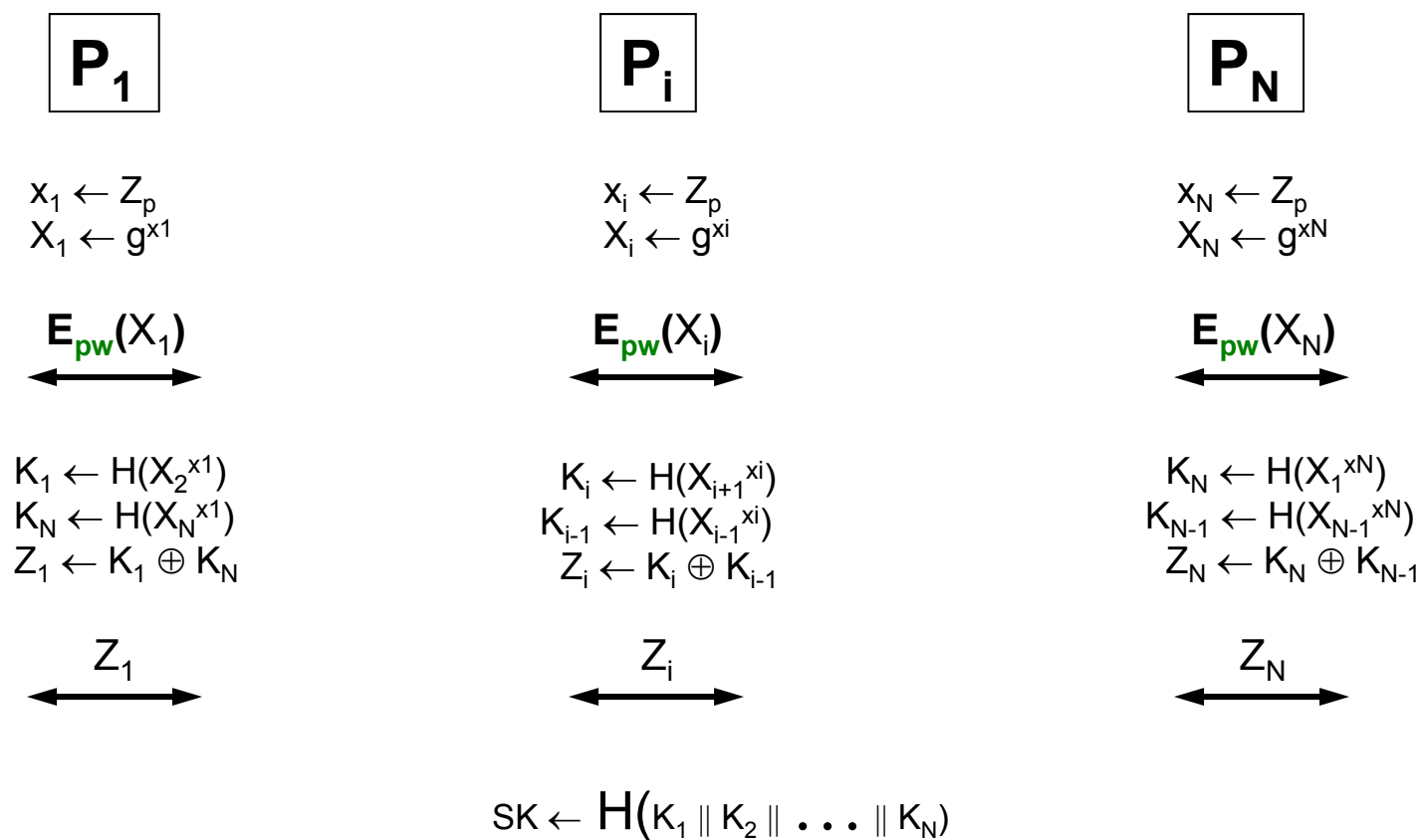
- **Attack**

- Let P_1 and P_2 be honest users
- Attacker will play the role of P_3
- Attacker waits for P_1 and P_2 to broadcast $X_1^* = E_{pw}(X_1)$ and $X_2^* = E_{pw}(X_2)$
- Attacker sets $X_3^* = X_1^*$ (This implicitly sets $x_1 = x_3$) and broadcasts it
- This causes $K_1 = K_2$ and $Z_2 = 0$
- Hence, $T_2^* = E_{pw}(0 || s_2) \Rightarrow$ Dictionary attack!

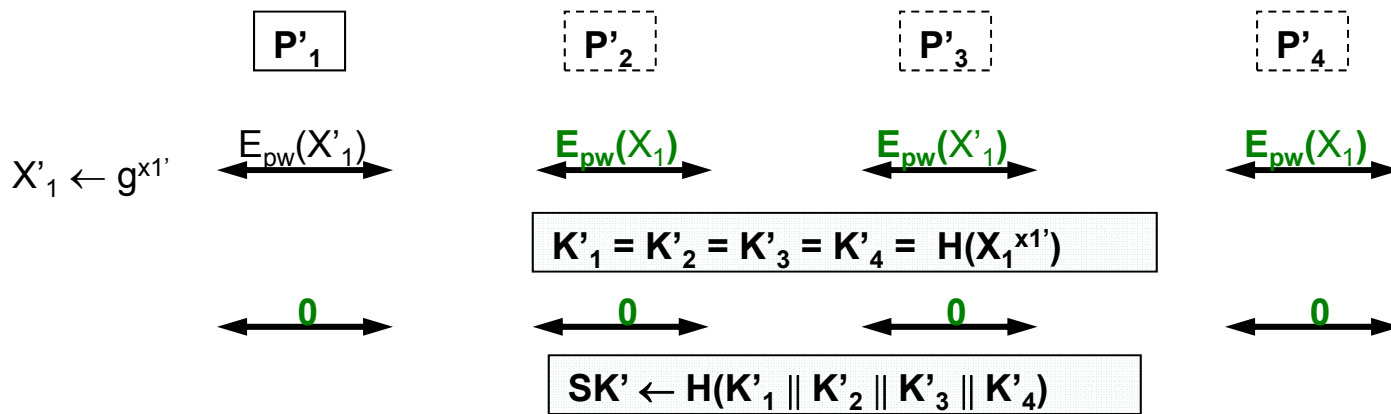
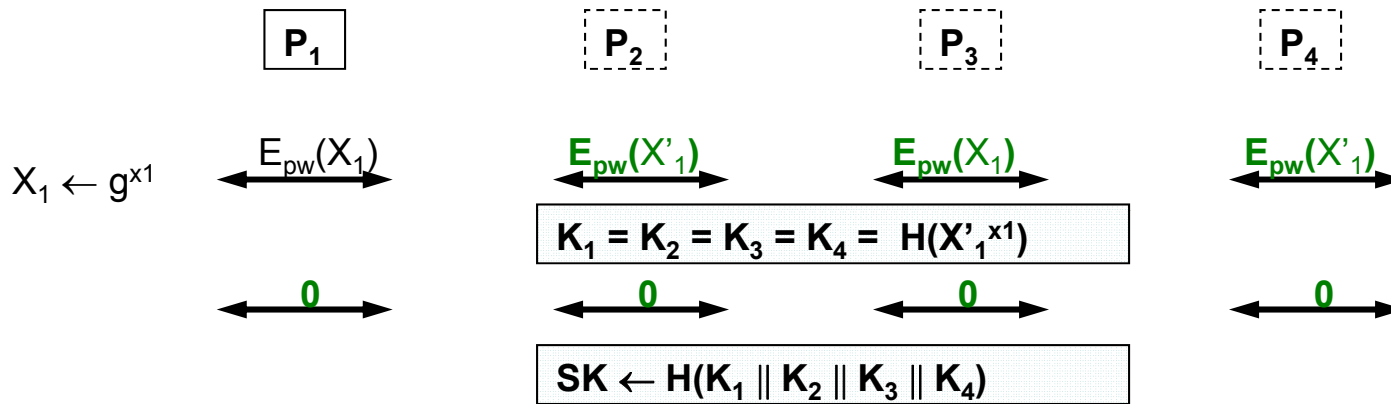
An attack against the Dutta-Barua GPAKE protocol



Lee-Hwang-Lee GPAKE [LHL04]



An attack against the Lee-Hwang-Lee GPAKE protocol



A simple GPAKE protocol

- Add an **extra flow of random nonces** r_i at the beginning of the each session

$$S = P_1 \parallel r_1 \parallel \dots \parallel P_N \parallel r_N$$

- Use a different encryption key for each user and session to avoid replaying of messages

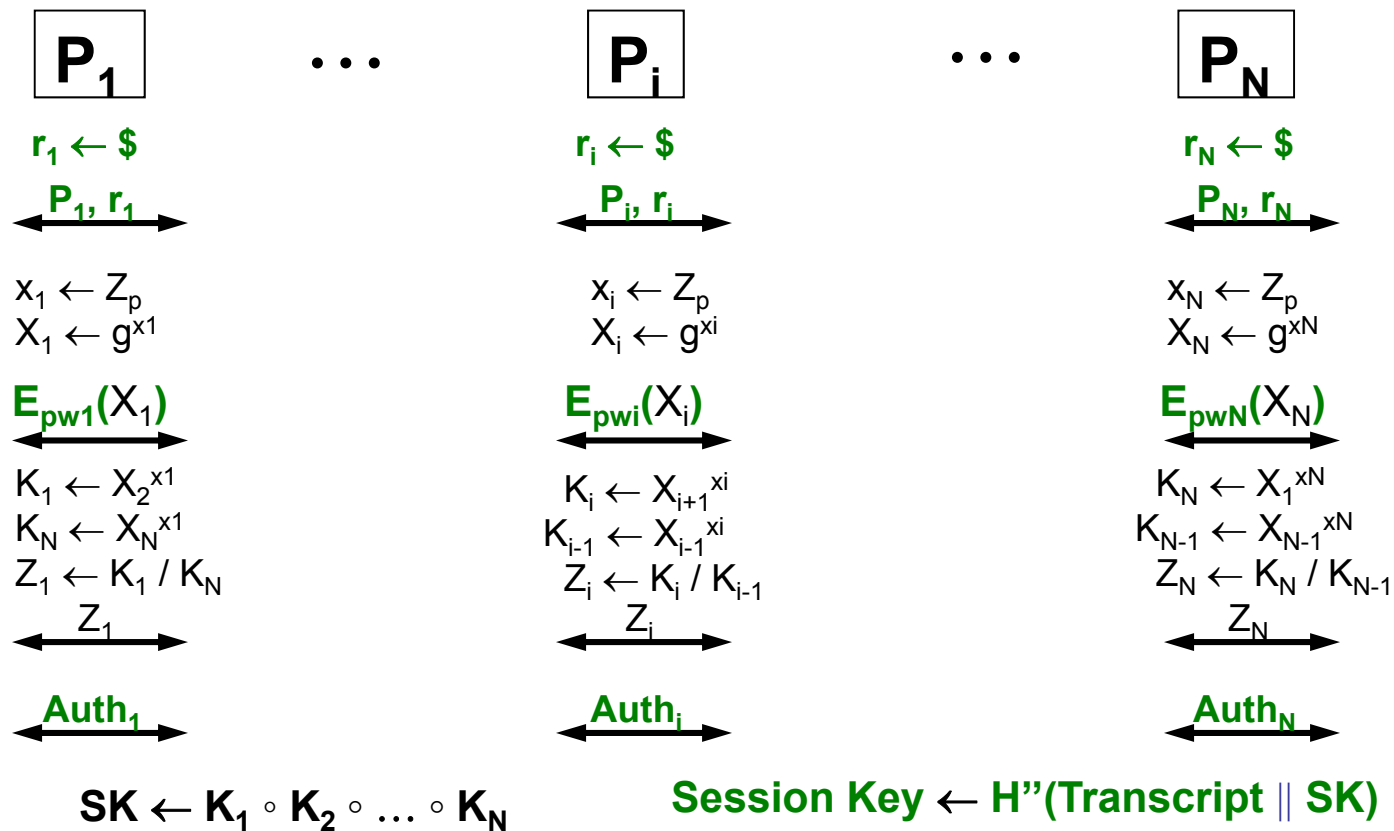
$$pw_i = H(pw \parallel S \parallel i)$$

- Only encrypt the flow containing the values X_i to avoid dictionary attacks

- Add an authentication flow to avoid malleability attacks

$$Auth_i = H'(S \parallel X_1^* \parallel Z_1 \parallel \dots \parallel X_N^* \parallel Z_N \parallel SK \parallel i)$$

A simple GPAKE protocol: Construction [ABCP06]



Security of ABCP06 GPAKE

- If the **DDH** problem is hard, then the protocol described in the previous slide is a **secure GPAKE** protocol in the random-oracle and ideal-cipher models

Outline

- ✓ Security model
- ✓ Unauthenticated key exchange protocols
- ✓ PKI-based authenticated key exchange
- ✓ ROM-based Password-based AKE
- PAKE in the standard model
- Security in the UC framework
- Concluding remarks

Background: Smooth projective hash functions [GL03 variant]

- An SPHF H is a family of functions that can be computed in two ways
 - Using the **secret hash key**
 - This is true for all points x of the domain X
 - Using a **public projected key**
 - This is only possible for points x in a particular subset $L \subset X$
 - The computation requires the knowledge of a witness w to the fact that $x \in L$

SPHF: Overview

- Let
 - $H: K \times X \rightarrow \Pi$ be a family of functions from X to Π , indexed by K
 - L be a **non-empty proper subset** of X
 - $\alpha: K \rightarrow S$ be a **projection function** from K to S
- For any point $x \in X$ and $k \in K$,
 - $H(k,x)$ can be easily computed given x and k
- For any point $x \in L$,
 - $H(k,x)$ can be **alternatively** computed given x , $\alpha(k)$, and a witness to the fact that $x \in L$

SPHF: Algorithms

- **Hash key generation:** $k = \text{Hash-KG}(L)$
 - k – hashing key, L – associated language,
- **Projected key generation:** $hp = \alpha(k)$
 - k – hashing key, hp – projected key
- **Hashing algorithm:** $y = H(k, x)$
 - x – input, y – output, k – hashing key
- **Projected hashing algorithm:** $y = h hp, x; r$
 - hp – projected key, r – witness for $x \in L$

SPHF: Security properties

- **Correctness**

- If $x \in L$, then $(x, \alpha(k))$ uniquely determines $H(k,x)$
- If $x \in L$, then $H(k,x)$ can be computed given $x, \alpha(k)$, and a witness to the fact that $x \in L$

- **Smoothness**

- If $x \notin L$, then $(x, \alpha(k))$ gives no information (statistically) on $H(k,x)$

- **Pseudorandomness:**

- If $x \in L$ and L is a hard-partitioned subset of X , then $H(k,x)$ is indistinguishable from random given only $(x, \alpha(k))$

A DDH-based SPHF example

- Let
 - \mathbf{G} be a cyclic group of prime order q with generators h_1, h_2
 - $X = (\mathbf{G} \times \mathbf{G})$; $\Pi = \mathbf{G}$; $K = (\mathbf{Z}_q, \mathbf{Z}_q)$; $S = \mathbf{G}$
 - L be the set of DDH tuples $x = (h_1^r, h_2^r)$
- **Hash key generation (Hash-KG)**
 - $k = (k_1, k_2)$ where $k_i \leftarrow \mathbf{Z}_q$
- **Hash computation (H)**
 - $H((k_1, k_2), (x_1, x_2)) = x_1^{k_1} x_2^{k_2}$
- **Projection function (α)**
 - $\alpha(k_1, k_2) = h_1^{k_1} h_2^{k_2}$
- **Hash computation with projected key (h)**
 - r is a witness to the fact that $x = (h_1^r, h_2^r) \in L$
 - $H((k_1, k_2), (x_1, x_2)) = (h_1^r)^{k_1} (h_2^r)^{k_2} = \alpha(k_1, k_2)^r$

Security properties of DDH-based SPHF

- **Correctness**

- $\mathbf{H}((k_1, k_2), (x_1, x_2)) = (h_1^{r'})^{k_1} (h_2^{r'})^{k_2} = (h_1^{k_1} h_2^{k_2})^{r'} = \alpha(k_1, k_2)^{r'}$

- **Smoothness**

- Let $hp = \alpha(k_1, k_2) = h_1^{k_1} h_2^{k_2}$
 - Since $x \notin L$, $x = (h_1^{r'}, h_2^{r'})$ for $r \neq r'$
 - Let $h' = \mathbf{H}((k_1, k_2), x)$ be the output of hash function \mathbf{H}
 - One can see that, for every choice of h' and hp' , there exists a pair (k_1, k_2) such that $\mathbf{H}((k_1, k_2), x) = x_1^{k_1} x_2^{k_2}$

- **Pseudorandomness**

- Follows easily from the DDH assumption

A Linear-based SPHF example

- Let
 - \mathbf{G} be a cyclic group of prime order q with generators h_1, h_2, h_3
 - $X = (\mathbf{G} \times \mathbf{G} \times \mathbf{G})$; $\Pi = \mathbf{G}$; $K = (\mathbf{Z}_q, \mathbf{Z}_q, \mathbf{Z}_q)$; $S = \mathbf{G} \times \mathbf{G}$
 - L be the set of Linear tuples $x = (h_1^{r_1}, h_2^{r_2}, h_3^{r_1+r_2})$
- **Hash key generation (Hash-KG)**
 - $k = (k_1, k_2, k_3)$ where $k_i \leftarrow \mathbf{Z}_q$
- **Hash computation (H)**
 - $H((k_1, k_2, k_3), (x_1, x_2, x_3)) = x_1^{k_1} x_2^{k_2} x_3^{k_3}$
- **Projection function (α)**
 - $\alpha(k_1, k_2, k_3) = (\alpha_1, \alpha_2) = (h_1^{k_1} h_3^{k_3}, h_2^{k_2} h_3^{k_3})$
- **Hash computation with projected key (h)**
 - r_1 and r_2 are witnesses to the fact that $x = (h_1^{r_1}, h_2^{r_2}, h_3^{r_1+r_2}) \in L$
 - $H((k_1, k_2, k_3), (x_1, x_2, x_3)) = (h_1^{r_1})^{k_1} (h_2^{r_2})^{k_2} (h_3^{r_1+r_2})^{k_3} = \alpha_1^{r_1} \alpha_2^{r_2}$

Cramer-Shoup encryption

[CraSho98]

- Let \mathbf{G} be a cyclic group of prime order q with generators g_1, g_2 and let H be universal one-way hash function

KG (pars)

$x_1, x_2, y_1, y_2, z_1, z_2 \leftarrow \mathbb{Z}_p$
 $e \leftarrow g_1^{x_1} g_2^{x_2}; \quad f \leftarrow g_1^{y_1} g_2^{y_2}; \quad h \leftarrow g_1^{z_1} g_2^{z_2}$
Return $(pk=(e,f,h), sk=(x_1, x_2, y_1, y_2, z_1, z_2))$

Enc (pars, (e,f,h) , m)

$u \leftarrow \mathbb{Z}_p^*$
 $a_1 \leftarrow g_1^u; \quad a_2 \leftarrow g_2^u$
 $c \leftarrow h^u \circ m$
 $v \leftarrow H(a_1, a_2, c)$
 $d \leftarrow (e f^v)^u$
 $C \leftarrow (a_1, a_2, c, d)$

Dec (pars, (e,f,h) , $(x_1, x_2, y_1, y_2, z_1, z_2)$, C)

$(a_1, a_2, c, d) \leftarrow C$
 $v \leftarrow H(a_1, a_2, c)$
 $m \leftarrow c a_1^{-z_1} a_2^{-z_2}$
If $d \neq a_1^{x_1+y_1v} a_2^{x_2+y_2v}$ then $m \leftarrow \perp$
Return m

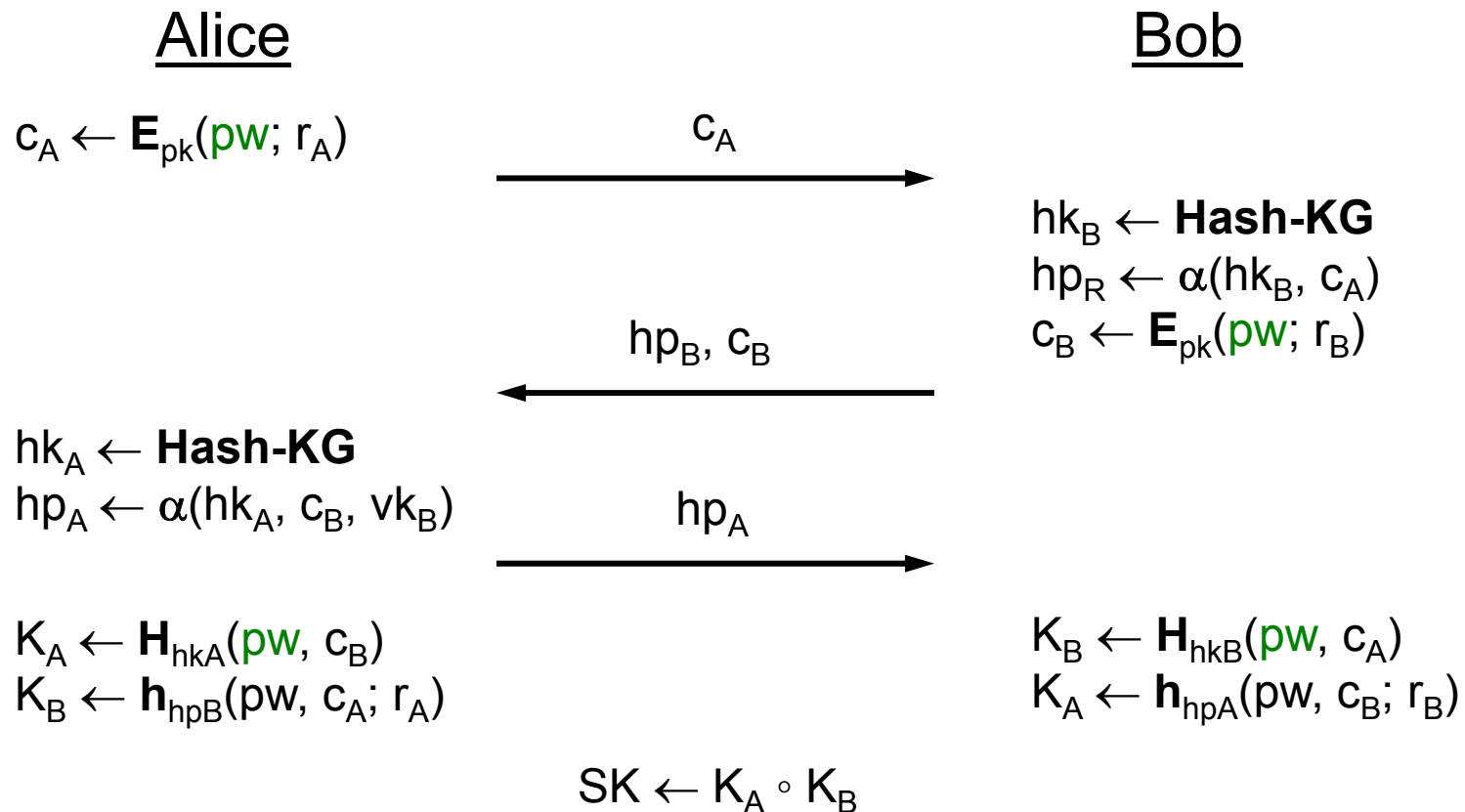
A Cramer-Shoup-based SPHF

- Let
 - \mathbf{G} be a cyclic group of prime order q with generators g_1, g_2
 - $X = \mathbf{G}^{4 \times \mathbf{G}}$; $\Pi = \mathbf{G}$; $K = \mathbf{Z}_q^4$; $S = \mathbf{G}$
 - $L = L_{CS} = \{ (C, m) \mid \exists r \text{ s.t. } C = CS_{pk}(m; r) \}$
- **Hash key generation (HK)**
 - $k = (k_1, k_2, k_3, k_4)$ where $k_i \leftarrow \mathbf{Z}_q$
- **Hash computation (H)**
 - $H(k, (C, m)) = a_1^{k_1} a_2^{k_2} (c/m)^{k_3} d^{k_4}$, where $C = (a_1, a_2, c, d)$
- **Projection function (α)**
 - $\alpha(k, (C, m)) = g_1^{k_1} g_2^{k_2} h^{k_3} (e f^v)^{k_4}$, where $v = H(a_1, a_2, c)$
- **Hash computation with projected key (h)**
 - r is a witness to the fact that $(C, m) \in L$ (i.e., $C = CS_{pk}(m; r)$)
 - $H(k, (C, m)) = (g_1^r)^{k_1} (g_2^r)^{k_2} (h^r)^{k_3} ((ef^v)^r)^{k_4} = \alpha((k_1, k_2), (C_1, C_2))^r$

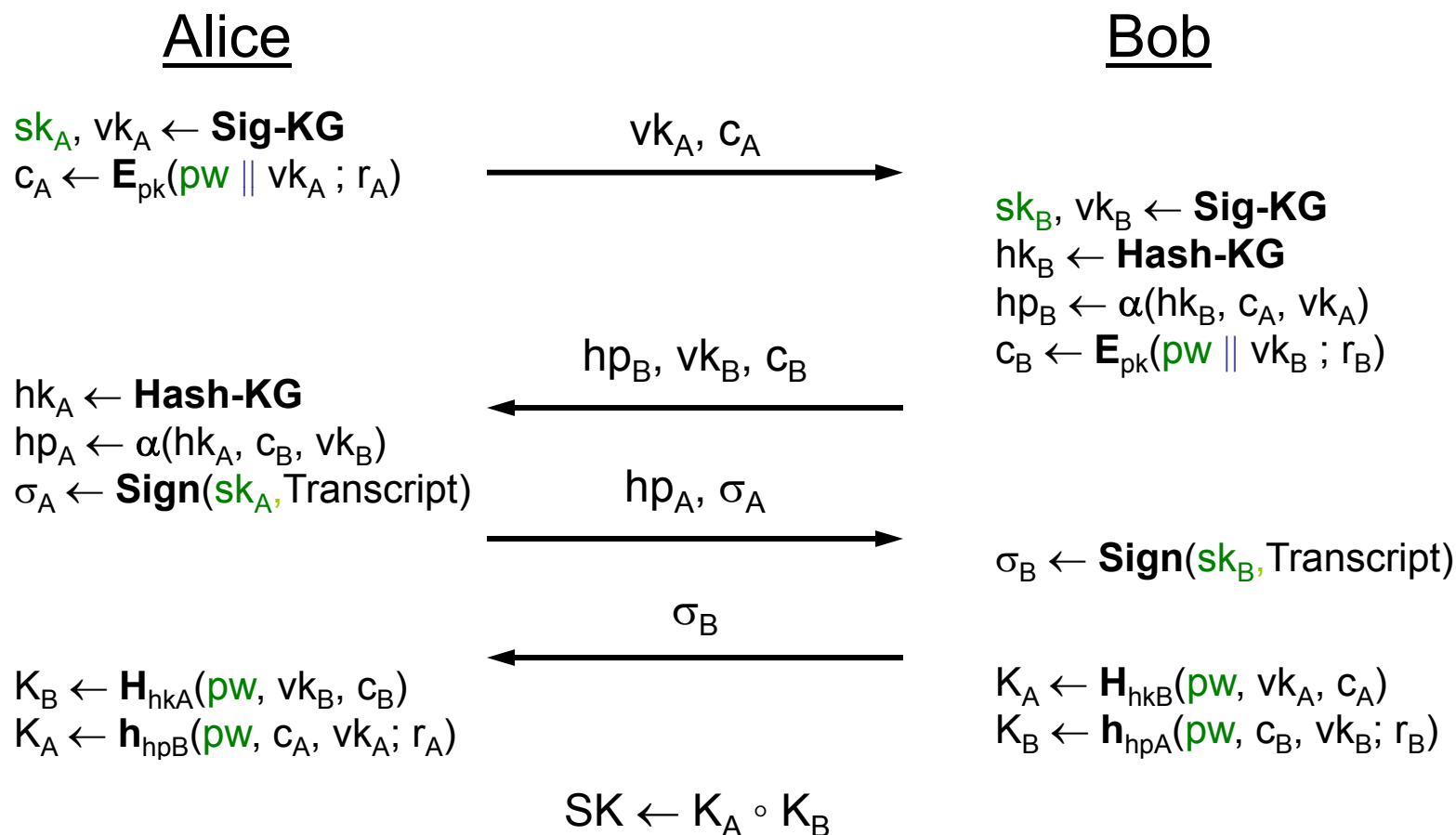
Gennaro-Lindell/KOY PAKE

- Basic tools:
 - **E**: IND-CCA PKE scheme
 - **H**: SPHF for the language $L_{\text{enc}} = \{\langle c, m \rangle \mid \exists r \text{ s.t. } c = \mathbf{E}(\text{pk}, m; r)\}$
 - **Sig**: a one-time signature scheme
- Observation:
 - The GL/KOY scheme uses a more general definition of SPHF in which the projection key may also depends on the ciphertext

Gennaro-Lindell/KOY PAKE: Main idea



Gennaro-Lindell/KOY PAKE



Security of GL/KOY PAKE

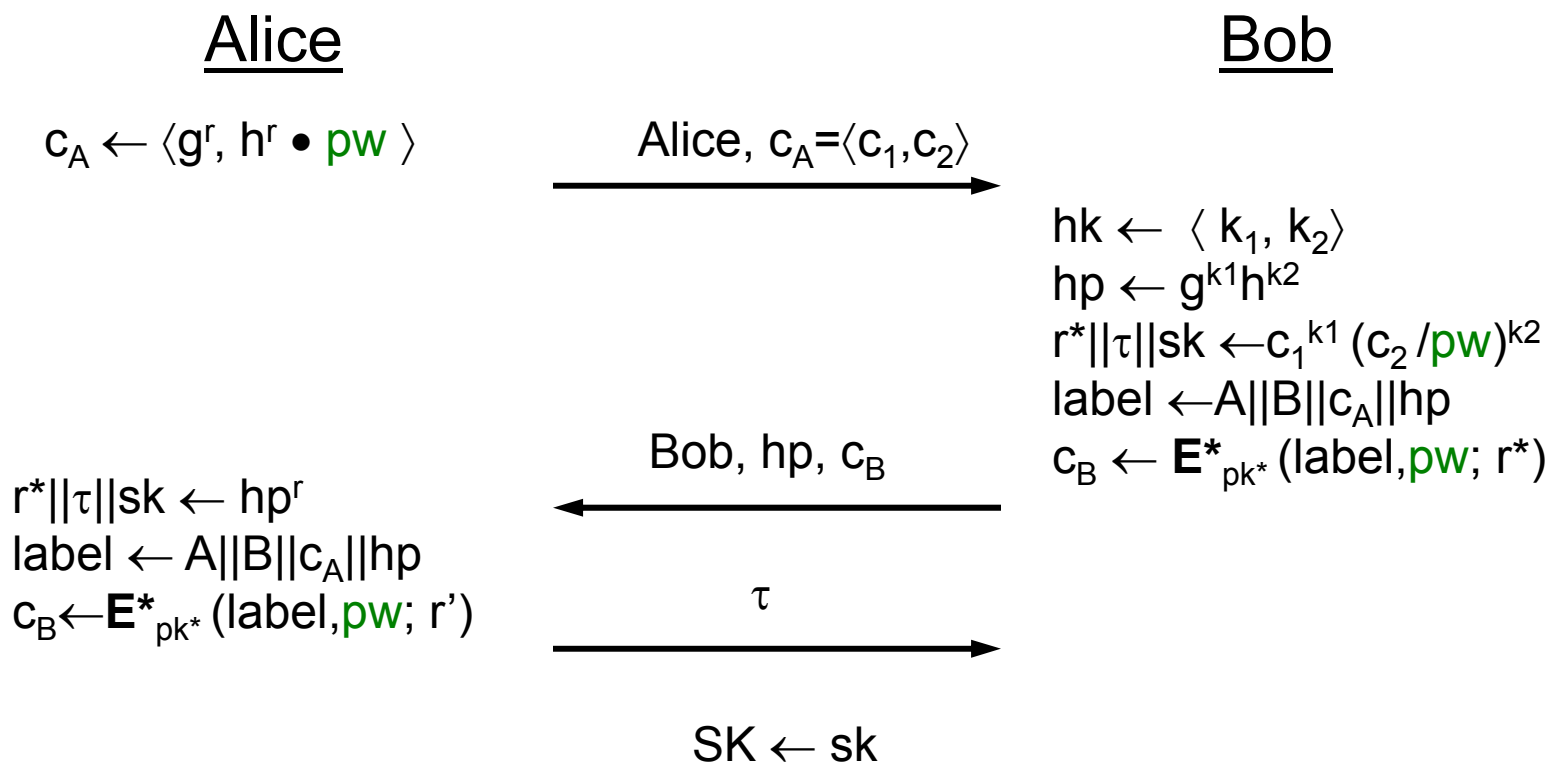
- **Passive adversaries**

- The **pseudorandom** property of the SPHF ensures that the session key will be indistinguishable from random

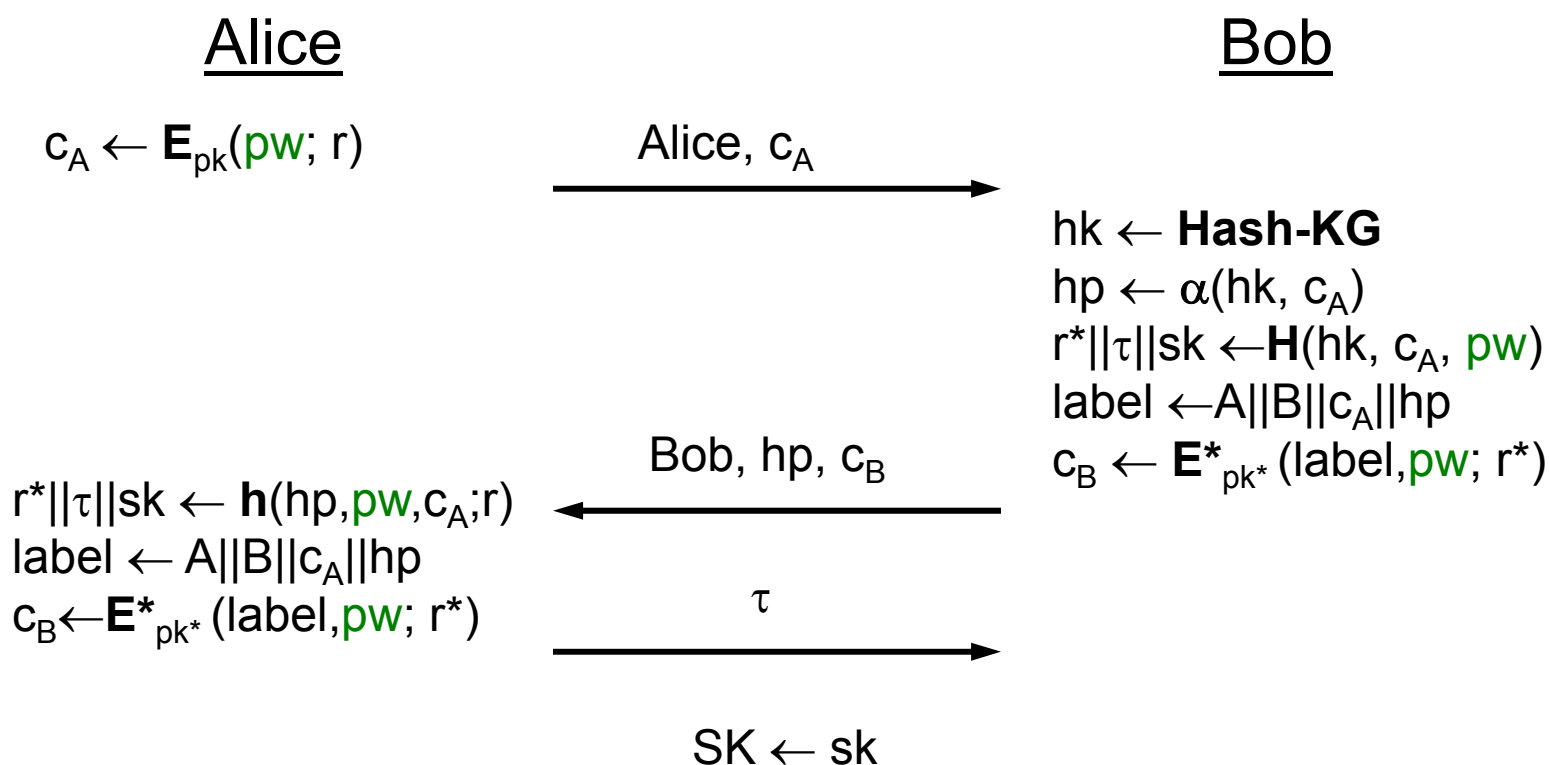
- **Active adversaries**

- If adversary provides with an **encryption of the wrong password**, the security of the session key follows from the **smoothness** property \Rightarrow adversary has to generate the encryption of correct password
- Since E_{pk} is IND-CCA, modifying existing ciphertexts is not possible
- *Reusing existing ciphertexts is not possible because either **vk would not match** or the **signature would be invalid***
- Remaining strategy: **guess the password**

Jiang-Gong PAKE

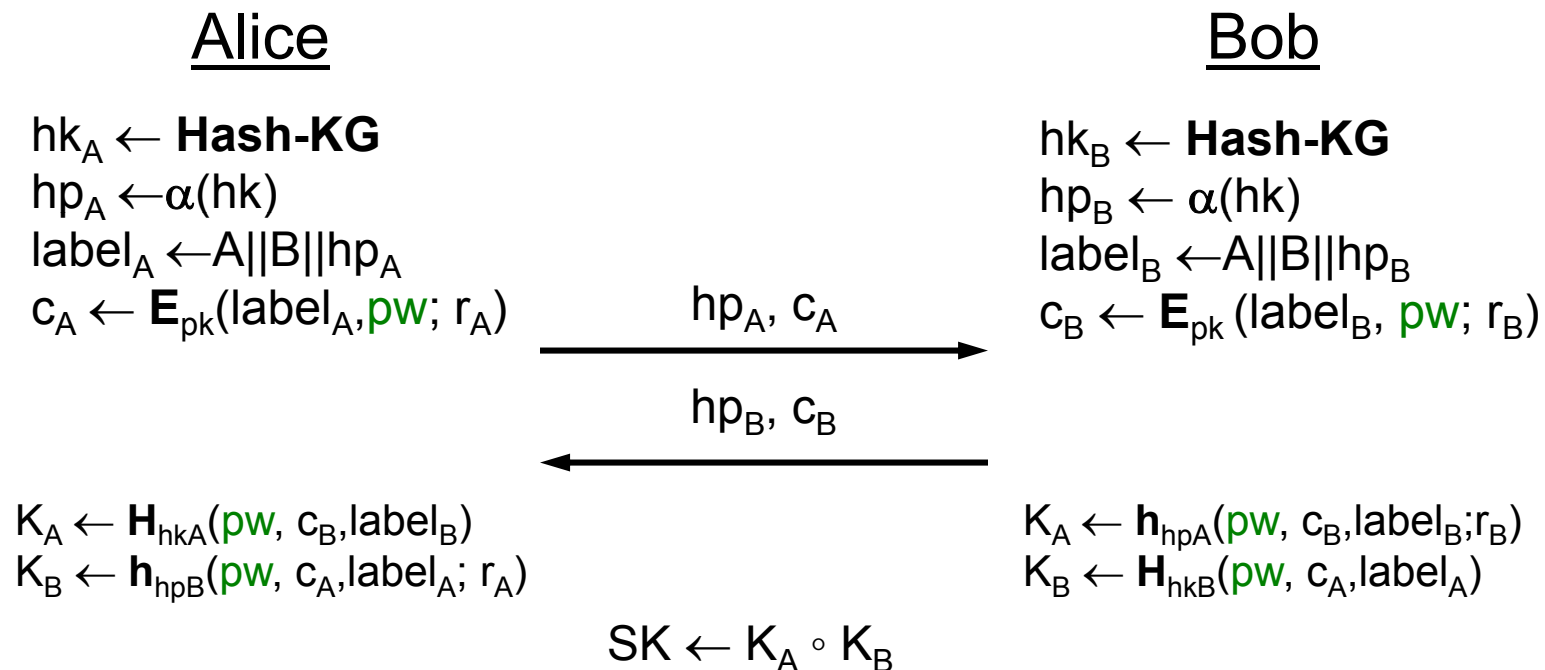


Groce-Katz PAKE



$L_{\text{enc}} = \{ \langle c, m \rangle \mid \exists r \text{ s.t. } c = \mathbf{E}(pk, m; r) \}$ where \mathbf{E} is IND-CPA

Katz-Vaikuntanathan PAKE



- **Observation:**

- $L_{enc} = \{ \langle label, c, m \rangle \mid \exists r \text{ s.t. } c = \mathbf{E}(label, pk, m; r) \}$ where \mathbf{E} is IND-CCA
- The projection key needs to be independent of the ciphertext

Security of KV PAKE scheme

If

- **E** is an CCA-secure labeled PKE scheme and
- **H** is a SPHF

Then

- KV is a **secure against** active attacks
- $\text{Adv}(A) \leq O(q_{\text{send}}/|\mathbf{Dictionary}|) + \text{negl}(k)$

Proof by games: G_0 to G_4

- G_0 : The real game
- G_1 : $K_A = \$$, $K_B = \$ H()$ in **execute** queries
 - Difference is negligible due to pseudorandomness of SPHF
- G_2 : $c_A = \mathbf{E}_{pk}(\text{label}_A, \mathbf{0})$ & $c_B = \mathbf{E}_{pk}(\text{label}_B, \mathbf{0})$ in **execute** queries
 - Distance is negligible due to semantic security of \mathbf{E}
- G_3 : if (hp_x, c_x) is prev. used, compute SK w/ hk_x in **send** queries
 - $\text{Adv}_3(A) = \text{Adv}_2(A)$
- G_4 : if $\mathbf{D}_{sk}(\text{label}_x, c_x) \neq \text{pw}$, then set $\text{SK} = \$$ in **send** queries
 - Distance is negligible due to smoothness of SPHF

Proof by games: G_5 to G_7

- **G_5** : if $D_{sk}(\text{label}_X, c_X) = \text{pw}$, then abort and declare A successful
 - $\text{Adv}_3(A) \geq \text{Adv}_2(A)$
- **G_6** : if (hp_X, c_X) is prev. used in **send** queries, then set $SK_X = \$$ if no partner or $SK_X = SK_{\text{Partner}}$
 - Distance is negligible due to IND-CCA security of E
- **G_7** : $c_A = E_{pk}(\text{label}_A, \mathbf{0})$ & $c_B = E_{pk}(\text{label}_B, \mathbf{0})$ in **send** queries
 - Distance is negligible due to IND-CCA security of E
 - $\text{Adv}_7(A) \leq O(q_{\text{send}}/|\mathbf{Dictionary}|)$

Outline

- ✓ Security model
- ✓ Unauthenticated key exchange protocols
- ✓ PKI-based authenticated key exchange
- ✓ ROM-based Password-based AKE
- ✓ PAKE in the standard model
- Security in the UC framework
- Concluding remarks

Shortcomings of existing security models

- Passwords are chosen from pre-determined, known distributions
- Passwords shared between different parties are assumed to be independent
- No security guarantees under arbitrary compositions

Composition of multiple protocols

- During reduction, the simulator has to emulate the environment until the adversary wins its game
 - What about the composition of multiple protocols?
 - Simulation may fail as soon as an adversary breaks one of the parts of the global system, even though other parts may still provide protection
 - Other executing protocols may provide additional information to the adversary
- ⇒ Either re-prove the entire system or prove each component in the **UC Framework**

Implications of UC

- Can design and analyze protocols in a modular way:
 - Partition a given task T to simpler sub-tasks $T_1 \dots T_k$
 - Construct protocols realizing $T_1 \dots T_k$
 - Construct a protocol for T assuming ideal access to $T_1 \dots T_k$
 - Use the composition theorem to obtain a protocol for T from scratch.
- ⇒ Now can be done concurrently and in parallel

Security in the UC framework [CHKLM05]

- **A new definition in UC framework**
 - No assumption on the distribution of passwords
 - Passwords can be related
 - Security under arbitrary composition
- **A new protocol**
 - Generalization of GL/KOY protocol
 - Security proof based on standard complexity assumptions
 - Not as efficient as existing protocols in BPR/BPM model

Outline

- ✓ Security model
- ✓ Unauthenticated key exchange protocols
- ✓ PKI-based authenticated key exchange
- ✓ ROM-based Password-based AKE
- ✓ PAKE in the standard model
- ✓ Security in the UC framework
- Concluding remarks

Concluding remarks

- **The design of key exchange protocols can be tricky**
 - Small modifications to the protocol can make them insecure
 - The only way to be sure is to provide a security proof
- **What's the correct security model?**
 - Though we used a particular one in these lectures, there are several proposals in the literature
 - In the case of PAKE, UC secure definitions seem to provide stronger security guarantees (but protocols are less efficient)
- **Authenticated key exchange remains a very active area**
 - Attacks against previous schemes
 - New security models
 - More efficient protocols