An Introduction to the Design and Analysis of Key Exchange Protocols

(Bar Ilan Winter School – Feb 2018)

Hugo Krawczyk

IBM Research

hugo@ee.technion.ac.il

webee.technion.ac.il/~hugo/KE1_Hugo_BIU_Feb2018-online.pdf

About this class

- A mostly-informal introduction (especially the first half)
- Focus on design principles, conceptual understanding
  - Examples from real world, standardized key exchange protocols but at a high level ("cryptographic core")
  - Simplicity as a "core value"
  - Learning by counter-example
  - "undergraduate level"
- Formalisms and details in coming days
- Note on terminology: "Key exchange" = "Authenticated Key Exchange" (also, key agreement)

Two Major Sins

1. Mostly based on my own works
   - Recycling talks
2. Worse: I think it is ok
   - Not intended as a comprehensive survey (need a full+ semester for that) but as a conceptual introduction

Part I

- Introduction
  - What is a Key Exchange Protocol
  - Formalization and Design Challenges
- Main examples: Authenticated Diffie-Hellman
- Common pitfalls and sound design principles
  - Learning by examples and counter-examples
- Formalizing and proving key exchange protocols
  - Modular design and analysis: Authenticators
- Protocols
  - ISO and SKEME
  - STS and Photuris
  - SIGMA Protocol and identity privacy
- IPsec's Internet Key Exchange (IKE)

Part II

- Implicit authenticated key exchange
  - In the complex search for extreme simplicity
  - HMQV, Okamoto-Tanaka, One-Pass KE

Part III

- Key derivation & the HKDF extract-and-expand scheme

What is a Key Exchange Protocol

- Many answers: From naïve intuition to rigorous formal theory
- The fundamental role of KE
  - Bootstrap a secure channel between two communicating parties via the negotiation of shared keys and cryptographic services
  - Enabler and heart of secure communications
  - The link between long-term keys and fresh session keys
    - Link between public-key and symmetric cryptography
- The most common form of cryptographic protocol in wide use
  - Can teach us a lot about design and analysis of more complex protocols
  - **Deceptively simple** (overlooked as a "given" in multi-party protocols)

Key Exchange Protocols
(very informal)

- A protocol between two parties to establish a shared key ("session key") such that:

1. Authenticity: they both know who the other party is
2. Secrecy: only they know the resultant shared key

Also crucial (yet easy to overlook):

3. Consistency: if two honest parties establish a common session key then both have a consistent view of who the peers to the session are

A: (B,K) and B: (id,K) ➔ id=A

Key Exchange Protocols

- More generally:
  - Multiple parties; any two may exchange a key
  - Sessions: multiple simultaneous executions
- Adversary:
  - Monitors/controls/modifies traffic (man-in-the-middle)
  - May corrupt parties: learns long-term secrets
  - May learn session-specific information: state/keys
- Security goal: preserve authenticity, secrecy and consistency of uncorrupted sessions
  - Confine damage from exposure to a minimum

Formalizing Key Exchange

- An intuitive notion but hard to formalize
- Wish list for formal definitions/model:
  - Intuitive (beware!)
  - Reject "bad" protocols (capture full capabilities of realistic attackers)
  - Accept "good", natural protocols (avoid overkill requirements)
  - Ensure security of KE applications: "secure channels" as the quintessential application (protocol composition)
  - **Usability**: easy to analyze (stand alone → composable)
    + a design tool

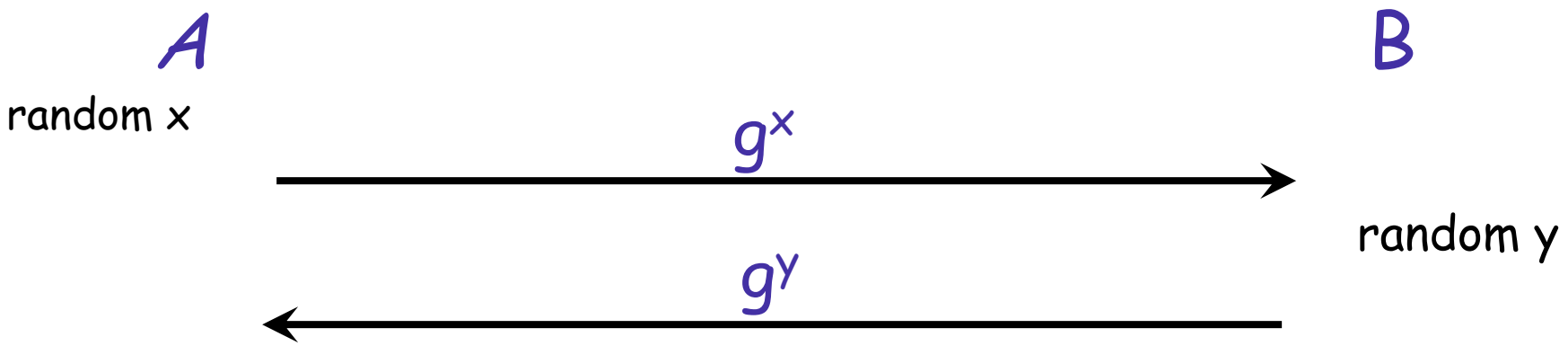
Designing and Analyzing KE Protocols…

- …is non-trivial
- Yet the end protocol need not be complex
  - And: to be practical the protocol *MUST BE SIMPLE*
- Best advice: learn from past experience (good and bad)
- Formal analysis as an indispensable tool
  - But remember: there is no ULTIMATE security model or absolute proofs of security
    - only relative to the model (and cryptographic assumptions)

Diffie-Hellman and Key Exchange Protocols

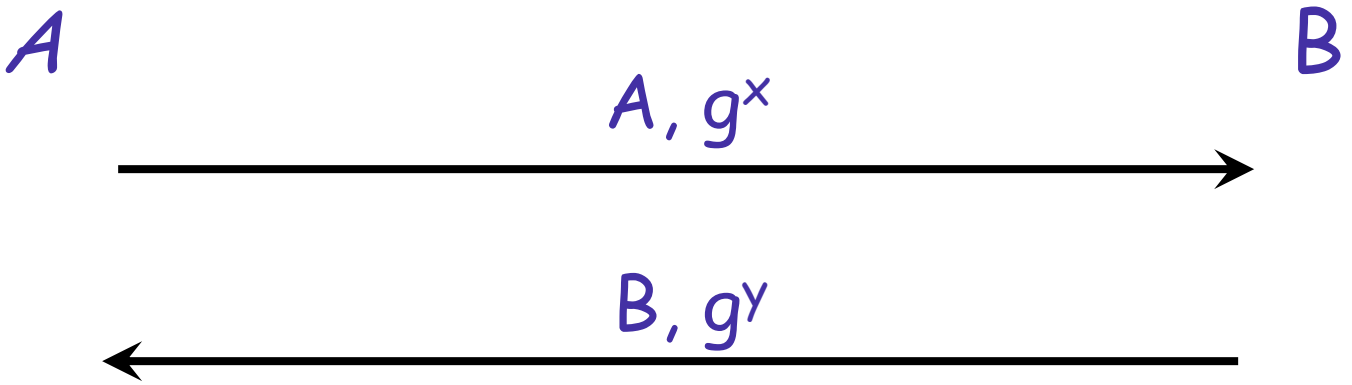
Diffie-Hellman Exchange [DH'76]

g = generator of a cyclic group G (e.g. Z_p^*, EC group)



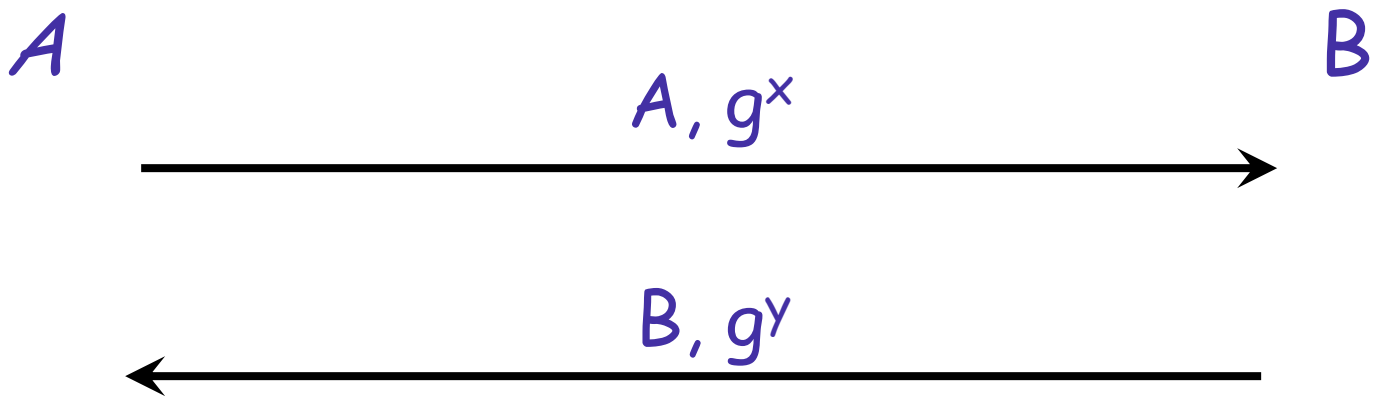
- Both parties compute the secret key $K=g^{xy}=(g^x)^y=(g^y)^x$
  1. CDH Assumption: K hard to compute given only g^x and g^y
  2. DDH Assumption: K indistinguishable from random element in G
     - Decisional DH assumption: $(g^x,g^y,g^{xy}) \approx_c (g^x,g^y,g^{random})$
- From g^{xy} to a k-bit key: KDF: $g^{xy} \rightarrow \{0,1\}^k$ (pseudorandom)

Diffie-Hellman KE and PFS



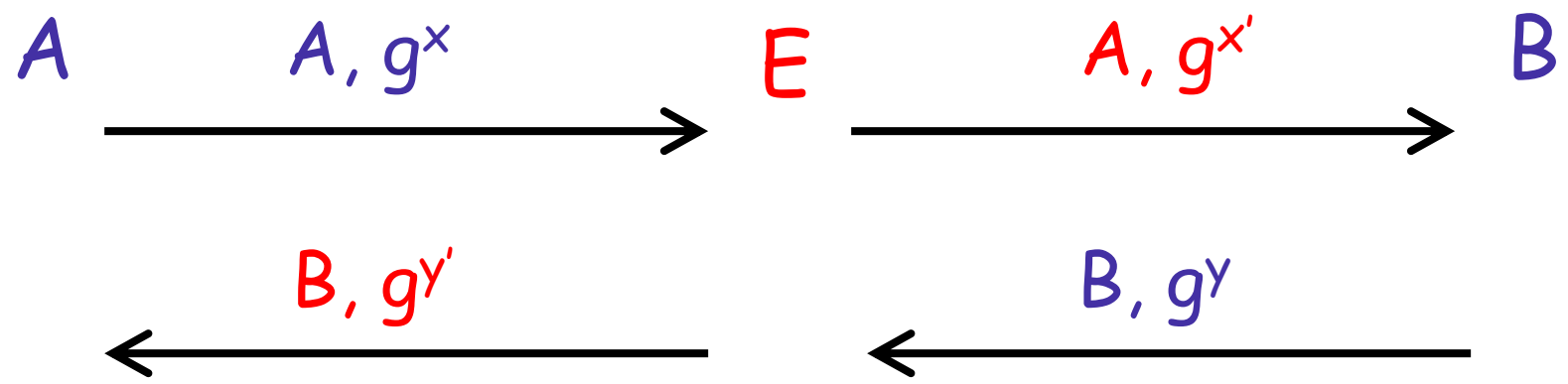
- Perfect Forward Secrecy (PFS)
  - Once the session keys are destroyed there is no way to recover them, not even by the owners (not even at gunpoint)
  - Distinguishes D-H from other protocols
    - compare SSL: What if your bank's private encryption key ever compromised? ALL past traffic exposed!
    - With PFS long-term keys used only for authentication

Diffie-Hellman Exchange [DH'76]



- beautiful, strong, but…
- secure only against eavesdroppers
- open to active attacker (man-in-the-middle)

(Wo)Man-in-the-Middle



$K_{AB}=g^{xy'}$

$K_{BA}=g^{x'y}$

Eve knows both keys!

The Long Journey Towards Authenticated DH Protocols

- UN-authenticated DH has survived to this date without change (40+ years!)
- In the same period, hundreds of papers published on *authenticated* DH protocols, *many (most?) of them broken!*
- Why is it so hard to get it right?

The Long Journey Towards Authenticated DH Protocols

- Why is it so hard?
  - What is authentication? Difficult notion, non-trivial to formalize. Changes with trust and setup scenarios.
  - How do we know when a protocol is secure? When are we done debugging it? Poor track record…
- Took long time to be able to develop sound, general security models
- And even more to prove protocols in these models
- And guess what… unproven protocols tend to be broken

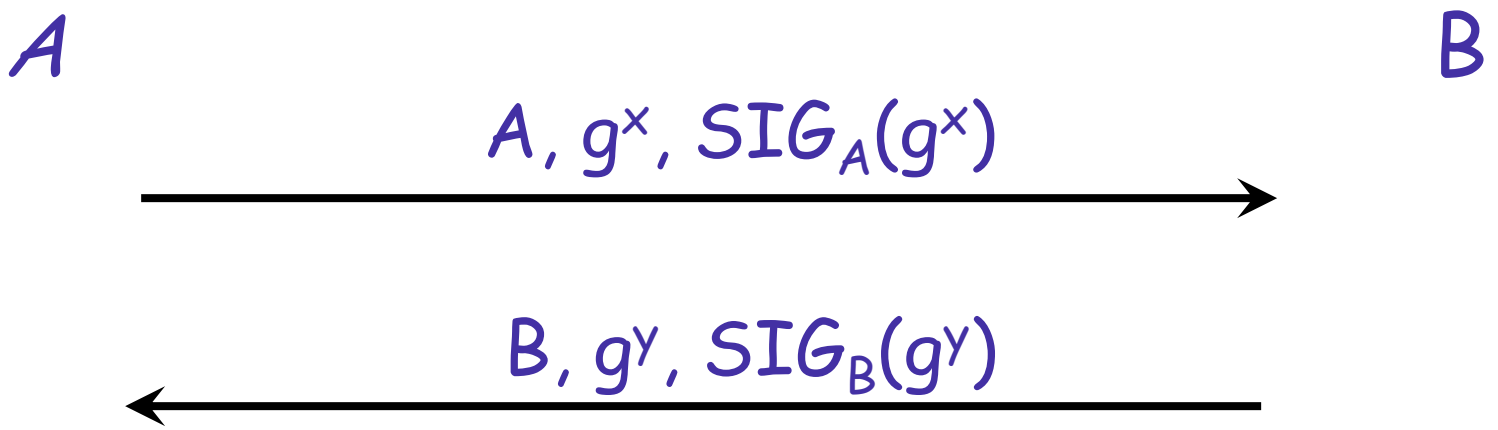
Guided Tour to

Authenticated DH Protocols

Conventions (and disclaimers)

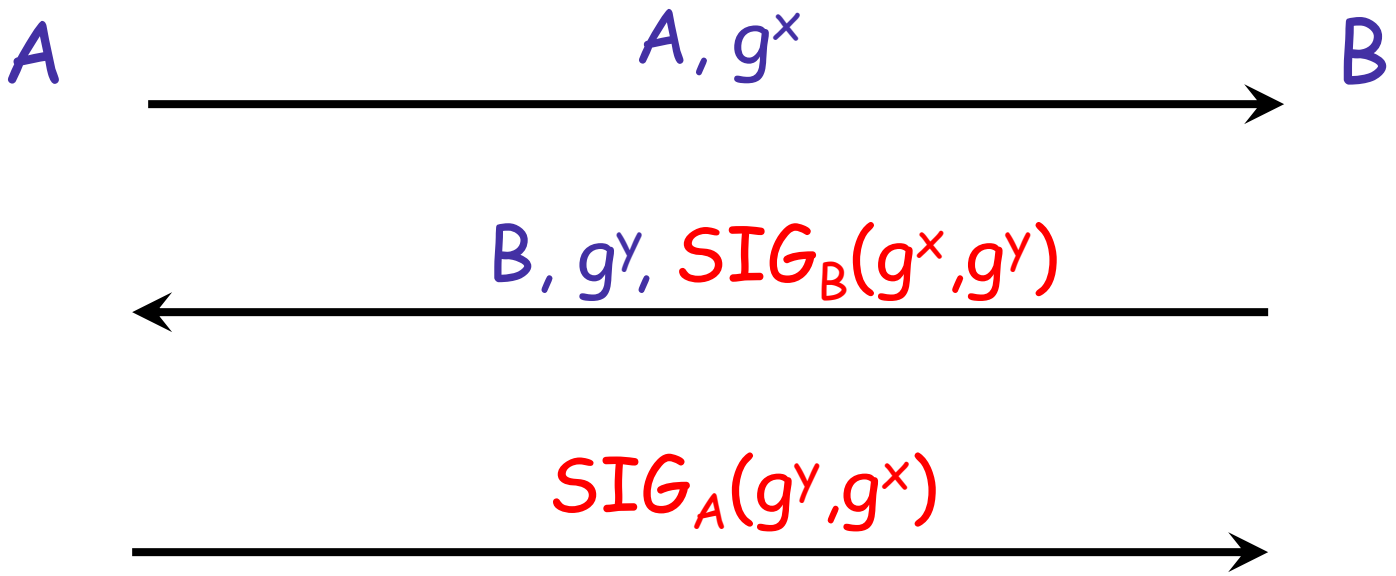
- A, B denote participants as well as their identities.
- Certificates: We assume parties have long-term public keys which other parties can learn and verify
  - Either by out-of-band verification or via certificates sent in the protocol (in the latter case A, B stand for identities plus certificates)
  - We omit many essential operations such as explicit verification operations (of signatures and certificates)
- DH group: We use "universal parameters" (e.g. p, g) – these can be wired into the protocol, negotiated during execution, etc.
- Note: No protocol can be secure without careful treatment of these "details" (remember: in crypto, the devil is in the details)
- Here the focus is on basic structure and general principles

First Attempt at Authenticated DH



- what if attacker ever finds a triple $(x, g^x, SIG_A(g^x))$?
  - E.g., file of precomputed (x, g^x) pairs
- Violates basic principle: Ephemeral leakage should not allow for long-term impersonation (beyond the leaked session)

Basic Authenticated DH ("BADH")



Each party signs its own DH value to prevent m-i-t-m attack

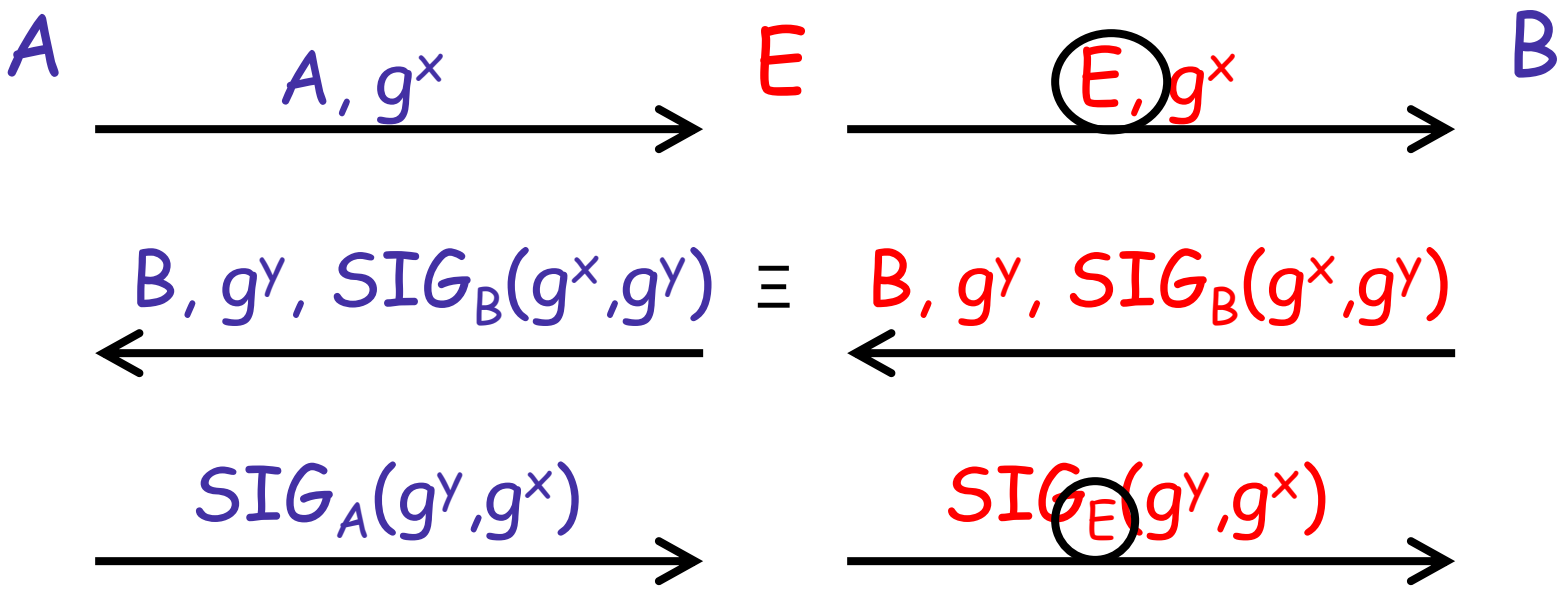
and the peer's DH value as a freshness guarantee against replay

A: "Shared K=g^{xy} with B" (K⇔B) B: "Shared K=g^{xy} with A" (K⇔A)

Looks fine, but…

Identity-Misbinding Attack [DVW'92]

(a.k.a. Unknown Key-Share attack = UKS)



- Any damage? Wrong identity binding!

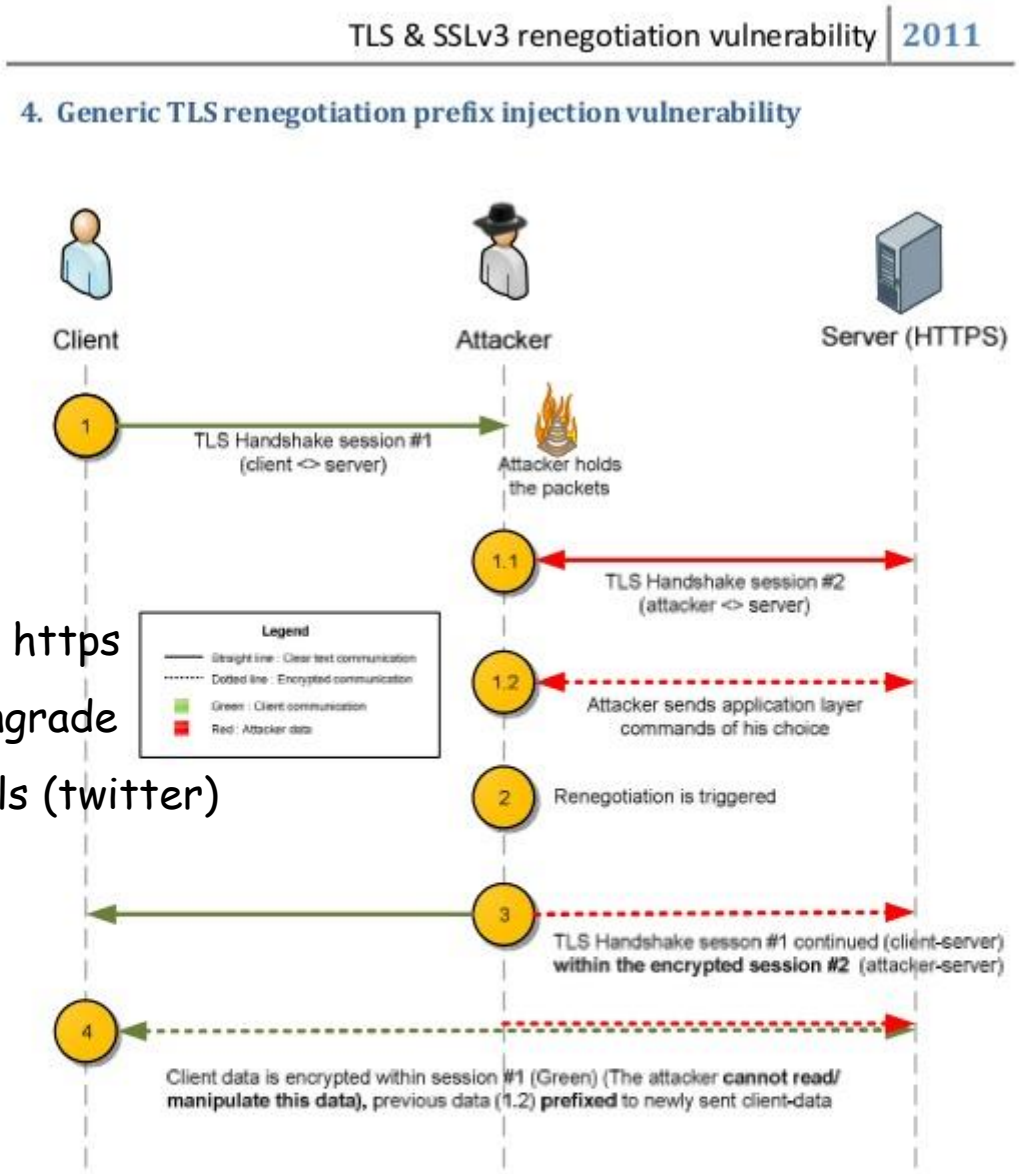
  A: "Shared $K=g^{xy}$ with B" ($K \Leftrightarrow B$) B: "Shared $K=g^{xy}$ with E" ($K \Leftrightarrow E$)

E doesn't know $K=g^{xy}$ but B considers anything sent by A as coming from E

A: "Shared $K=g^{xy}$ with B" (K⇔B)

B: "Shared $K=g^{xy}$ with E" (K⇔E)

- B = Bank A,E = customers
- A ⟶ B: {~~"deposit $1000 in my account"~~}$_K$
  - post this page on my website
  - posts the page
  - website
- B ~~deposits the money~~ in "K" 's ~~account~~, i.e. E's!
  - Should the bank protocol include explicit identities? Maybe, but KE should not make assumptions on higher-layer mechanisms
  - What is the expectation of higher layer protocols? That a key is uniquely bound to its owners ("speaks for its owners")
- SSL renegotiation's bug: wrong binding of sessions (attack succeeded without the attacker ever learning the key)



Some exploits:

- Command injection https
- https to http downgrade
- Stealing credentials (twitter)

www.slideshare.net/ThierryZoller/practicaltls1

Yet another example

HOME PAGE | TODAY'S PAPER | VIDEO | MOST POPULAR | U.S. Edition

The New York Times

Middle East

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS

AFRICA | AMERICAS | ASIA PACIFIC | EUROPE | MIDDLE EAST

Iranians Say They Took Secret Data From Drone

By STEVEN LEE MYERS and SCOTT SHANE
Published: April 22, 2012

WASHINGTON — Iran claimed on Sunday to have extracted secret intelligence information from an advanced American drone aircraft that crashed in the country in December, seeking a propaganda victory over what has been an embarrassing intelligence failure for the Central Intelligence Agency.

The air force commander of Iran's

Drone Example:

A: "Shared $K=g^{xy}$ with B" (K⇔B)

B: "Shared $K=g^{xy}$ with E" (K⇔E)

- A= F-16 B= Central Command E= drone (in enemy hands)
- B ⟶ E: $\{\text{"destroy yourself"}\}_K$
- E passes command $\{\text{"destroy yourself"}\}_K$ to A.
- Result: F-16 destroys itself!

Notes

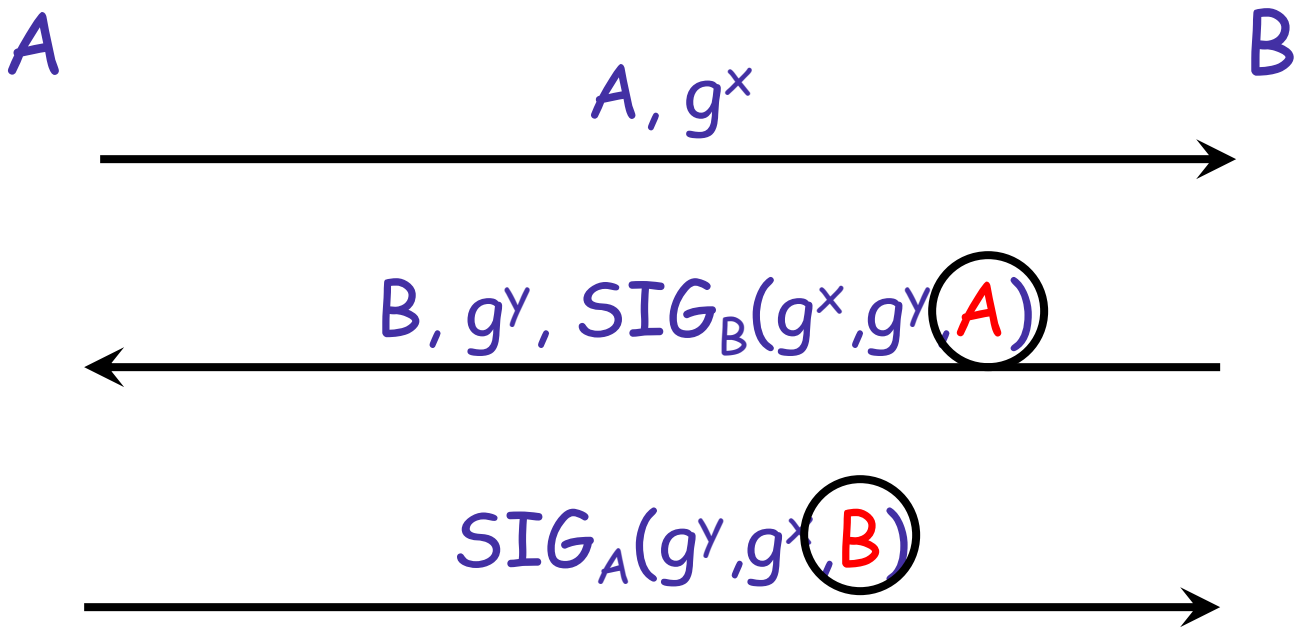
- Attack discovered by Diffie-van Oorschot-Wiener [DVW'92]
  - the "differential cryptanalysis" of KE protocols
  - highlights the crucial consistency property

    A: (B,K) and B: (id,K) ➔ id=A

- Note: The terminology Identity Misbinding Attack is mine

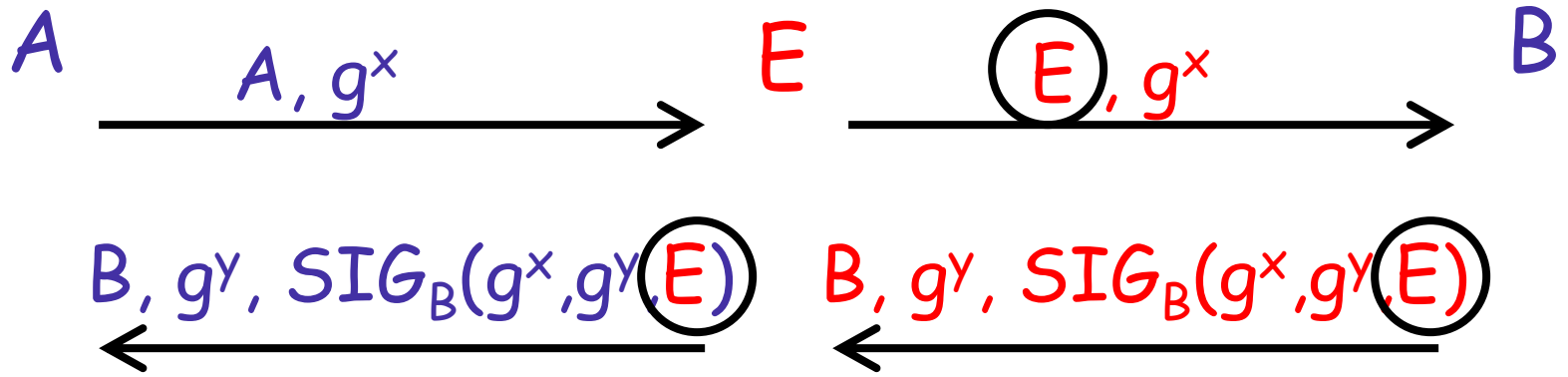
  The attack is more commonly referred to as the Unknown Key-Share (UKS) attack.

A Possible Solution (ISO-9796)



Thwarts the identity-misbinding attack by including the identity of the peer under the signature

The ISO defense



A: aha! B is talking to E not to me!

Note that E cannot produce $SIG_B(g^x,g^y,A)$

- The ISO protocol thus avoids the previous mentioned attacks; but is it secure??

The ISO Protocol is Secure

- We'll sketch the proof in the SK-security model of key exchange (known as the Canetti-Krawczyk model [CK'01])
- Note: the actual ISO-9796 protocol is more complicated: adds a MAC on the peers id -- which adds nothing to the security of the protocol
- An important consequence of well-analyzed protocols: avoiding "safety margins"
  - PROOF-DRIVEN DESIGN®

Let's then talk about KE models and proofs...

On KE Analysis Work

- Two main methodologies
  - Complexity based: security against computationally bounded attackers, proofs of security, reduction to underlying cryptography, probabilistic in nature
  - Logic-based analysis: abstracts crypto as ideal functions, protocols as state machines, good protocol debuggers
- Recent bridging work towards "the best of two worlds"
  - The power of automated analysis shown in recent TLS 1.3 design
- Here we focus on the first approach

Remember a "definition desiderata"

- Intuitive (session notion, attacker capabilities, secrecy)
- Reject "bad" protocols (capture full capabilities of realistic attackers)
- Accept "good", natural protocols (avoid overkill reqt's)
- Ensure security of KE applications: "secure channels" as the quintessential application + composition
- Usability: easy to analyze (stand alone → composable)
  + a design tool
- One more confidence source: Equivalence of different definitions

From Turing on Definitions of Computability

- A. Turing. On Computable Numbers, With an Application to the Entscheidungsproblem. *Proccedings of the London Mathematical Society*, 2(42):230–254, 1937.
- *No attempt has yet been made to show that the "computable" numbers include all numbers which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is "What are the possible processes which can be carried out in computing a number?"*
- *The arguments which I shall use are of three kinds.*
  a) *A direct appeal to intuition.*
  b) *A proof of the equivalence of two definitions (in case the new definition has a greater intuitive appeal).*
  c) *Giving examples of large classes of numbers which are computable.*

CK Model Predecessors

- Bellare-Rogaway'93
  - First complexity-theoretic treatment of KE
  - Indistinguishability approach [GM84]: attacker can't distinguish the real key from a random one
  - Extended in [BJM97] to the PK-authentication setting
- Bellare-Canetti-Krawczyk'98
  - Simulation-based definition of KE security
  - Ideally-authenticated (AM) vs. real-life (UM)
  - Modular authentication methodology
    - Authenticators: AM-to-UM compilers
- Both works required tunings (learning is a never-ending process)

Canetti-Krawczyk Model

- A combination of BCK'98 setting (simulation) and BR'93 indistinguishability approach ("SK-security")
  - Goal: ensure good composition and modularity properties but keep the simplicity of indisting'y-based analysis ("usability")
  - Later years: BR – CK convergence
- Secure channels as the "test application"
  - Requires a formalization of secure channels (e.g., a transport protocol such as IPSec, SSL, SSH) – simplifies, see KP talk
- Universally Composable security

SK Security [CK'01]

duplicate

- Geared towards allowing protocol composition [BCK'98] especially with generic "secure channels protocol"
- Follows indistinguishability approach [BR'93]
  - Defines a set of possible adversarial interventions/corruptions
  - Requires that such an attacker **cannot distinguish the shared session key from random** (as long as the parties to the session and the session itself are not corrupted)
- Secure channels as the must "test application"
  - Requires a formalization of secure channels (e.g., a transport protocol such as IPSec, SSL, SSH) – more later

SK-Security: KE protocol

- A two-party protocol in a multi-party setting
- Multiple protocol executions may run concurrently at the same or different parties
- Each run of the protocol at a party is called a *session* (a *local object*)
- Sessions have a unique local name: e.g. (A,s_A) and an incoming name (B,s_B) where B is the intended *peer.*
  The *session id* is the concatenation: (A,s_A,B,s_B)
- Sessions with corresponding names, i.e., (A,s_A,B,s_B) and (B,s_B,A,s_A) are called *matching*.
- Upon *completion* a session erases its state and outputs a triple: (session-id, peer-id, session-key)

SK-Security: Attacker

- Adversary model (Unauthenticated links Model - UM)
  - Full control of communication links: monitors/controls/modifies traffic (m-i-t-m)
  - Schedules KE sessions at will (interleaving)
  - May corrupt parties (total control): learns long-term secrets* (e.g. signature key), all its state and session keys
  - May issue a "learning query" for short-term information:
    - session state query (e.g., the exponent x of a g^x value)
    - session key query (of a complete, present or past, session)
- *Exposed session:* if session owner is corrupted, or attacker issued a query against the session, *or the matching session is exposed*
  - Clearly cannot protect a session if the matching is exposed

A KE Protocol is called SK secure if

1. Completed matching sessions output same session key (functional, non-triviality clause)
2. Attacker learns *nothing* about *unexposed* sessions
   - Captured via "test session"; chosen by attacker among completed *unexposed* sessions
   - Attacker is given either the session key or an independent random key; it needs to guess which one is the case
   - Require that the probability that the attacker guesses right is not significantly better than a random guess (i.e. ½ + small ε)

A compact but strong definition

- Captures many attacks that were *enumerated* in the past as separate requirements (or wish lists). For example:
  - Impersonation: if E can impersonate Bob without corrupting him then E knows a key of an unexposed session, contradicting the definition
  - Secrecy: If E learns anything about the session key then it can distinguish it from random. (Note: Why indistinguishability? The OTP example.)
  - Known-key attacks: An important class of attacks studied separately in the past: Can E break one session given the key of another session? Captured via session key query
  - Identity misbinding: if E forces two sessions w/outputs (A, B, K) and (B, E, K), then E can choose one as test and expose the other to learn K (it is allowed to do so since sessions are not matching)
- The definition can be further extended to cover other threats and security properties: e.g. PFS (via key expiration)

Informal Summary

- Summary of security guarantees for honest A,B:
  1. If A outputs session key (A, B, K) and B is honest then no one except B may know anything about K (not even a single bit)
     - Does the protocol guarantee that B outputs the key??
       - "key confirmation" possible but "common knowledge" is not
  2. Session keys are "computationally independent" of each other
     - Note: *session keys cannot be used during the key exchange itself* (cf. TLS)
- Note: Sharing a key with a corrupted party:
  - No guarantee for a key shared with a corrupted party.
  - But there is a guarantee that interacting with the attacker does not compromise any session between honest parties

Informal Summary

- Summary of security guarantees for honest A,B:
  1. If A outputs session key (A, B, K) and B is honest then no one except B may know anything about K (not even a single bit)
     - Does the protocol guarantee that B outputs the key??
       - "key confirmation" possible but "common knowledge" is not
  2. Session keys are "computationally independent" of each other
     - Note: *session keys cannot be used during the key exchange itself* (cf. TLS)
- Note: Sharing a key with a corrupted party:
  - No guarantee for a key shared with a corrupted party.
  - But there is a guarantee that interacting with the attacker does not compromise any session between honest parties

Beware! Triple Handshake Attack

About public keys

- Parties have long-term secrets
  - private signature/decryption keys, or shared keys w/other parties
- In the PK case: public keys of honest parties are assumed to be communicated to other parties correctly.
- Public keys of corrupted parties are chosen by the attacker arbitrarily (e.g., may be equal to a public key of another honest party).
  - Think of a CA that checks identity but no other properties of the keys being registered (does not assume/require proof of possession, checking structure of a key, etc.)

SK-security results (secure channels)

- SK-security ➔ Secure Channels

  authenticated encryption

  - Any key exchanged with an SK-secure KE protocol and used to **"encrypt-then-authenticate"** data realizes a secure channel [CK01]

- Secure channel realization:
  - K = output of KE (may require a KDF step, e.g. if $K=g^{xy}$; can be combined with prf step below)
  - Keys (K_{enc}, K_{mac}) = prf_K(context) where context may include session/protocol/algorithm identifiers, parties identities, etc.
    - Note: having directional keys (A→B and B→A) is a good idea
  - Apply to data: $c=Enc_{Kenc}(data)$, $t=Mac_{Kmac}(c)$; transmit (c,t)
- Other combinations, e.g. $Enc_{Kenc}(data \,||\, Mac_{Kmac}(data))$ may not be secure (TLS examples)

SK-security results (protocols)

- A variety of protocols have been proven SK-secure (both DH and key-transport)
  - e.g., ISO, IKE (SKEME, SIGMA), HMQV, Pre-Shared and more
  - Two SK-secure flavors: with and w/o PFS
    - PFS modeled through session-expiration (models erasure); expired sessions are NOT exposed even if attacker corrupts the session's owner.

SK-Security and Composition

- CK02: SK-Security is "universally composable" (UC [Can'02])
  - Remains secure under composition with any application, not just secure channels
  - Well, almost: true for protocols with the ACK property
  - True always if UC security weakened via "non-information oracles" (see CK02 eprint/2002/059)
- SK-Security preserved under *authenticators*
  - Authenticator: A "compiler" from (ideal) AM-secure protocols to (realistic) UM-secure protocols (a design and analysis tool!)
  - More in following slides (incl. the proof of the ISO protocol)

UM and AM Models

- Adversary model: UM (Unauthenticated links Model) AM (Authenticated links Model)
  - Full control of communication links: **monitors/controls/modifies** traffic (m-i-t-m) passive attacker
  - Schedules KE sessions at will (interleaving)
  - May corrupt parties (total control): learns long-term secrets (e.g. signature key), all its state and session keys
  - May issue a "learning query" for short-term information:
    - session state query (e.g., the exponent x of a g^x value)
    - session key query (of a complete, present or past, session)
- *Exposed session:* if session owner is corrupted, or attacker issued a query against the session, *or the matching session is exposed*
  - Clearly cannot protect a session if the matching is exposed

Authenticators [BCK98]

- Models:
  - UM (Unauthenticated-links Model): a realistic attack model as described before
  - AM (ideally Authenticated-links Model): like UM but attacker is passive; cannot change or inject msgs on links (but it may prevent delivery)
- ***Authenticator*** : a "compiler" from AM-secure protocols to UM-secure
  - Reduces the problem of designing (and analyzing) protocols from the complex UM to the simple AM
    - For example: Proving plain DH in the AM is immediate (under DDH assumption)

Authenticators (sketch of idea)

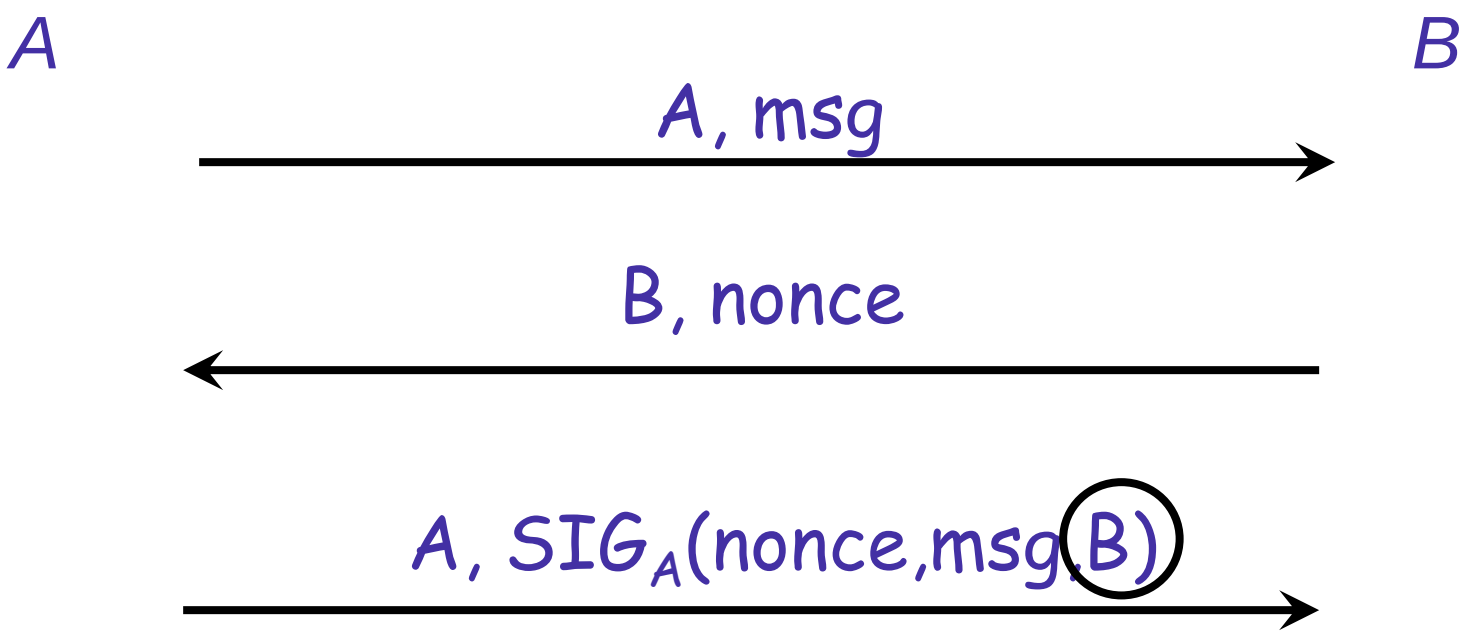
- Message sending protocol (can be interactive)
  - Parties send and receive messages and register their actions ("sent msg m to B", "received msg m from A")
- An *authenticator* is a message sending protocol s. t.
  - Whenever A registers "received m from B", it also holds that B registered "sent m to B"
  - Note: To prevent replay attacks messages need to be made unique (e.g., concatenated with msg id or nonce)

Authenticators Theorem

- Theorem: Let A be an authenticator
  - If P is a protocol secure in the AM model
  - and P' is the result of applying A to each message in P

  Then, P' is secure in the UM model.

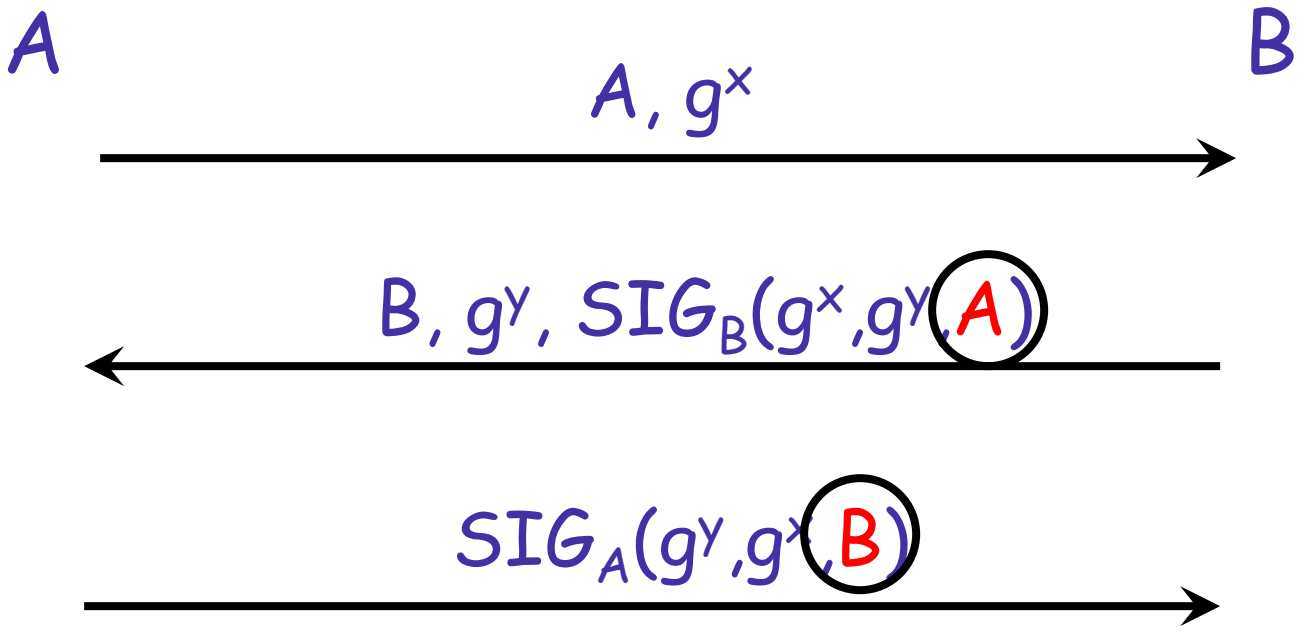
A signature-based authenticator
(applied to message msg)



Compiler from AM to UM: apply the above authenticator to each protocol's message

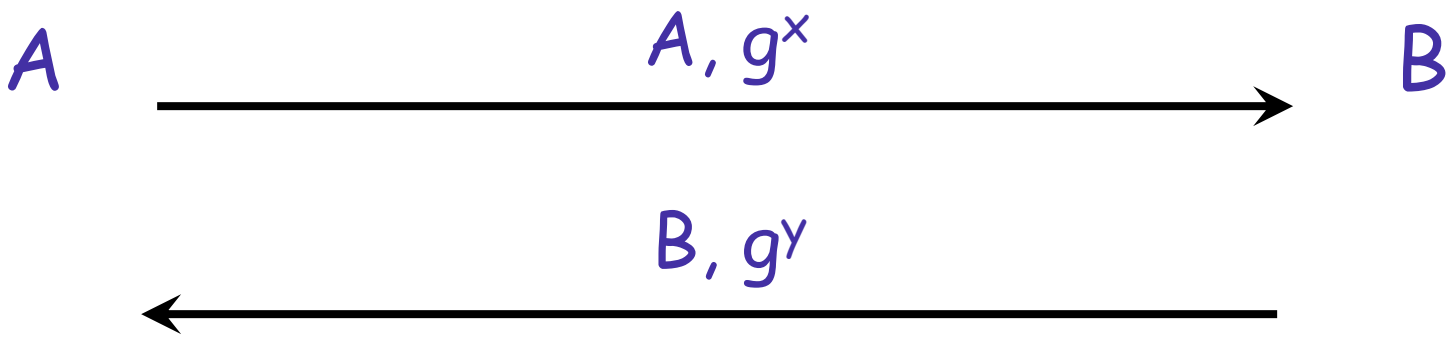
Proving ISO Using an Authenticator

Recall the ISO-9796 protocol
(solved the Identity-Misbinding Attack of BADH):



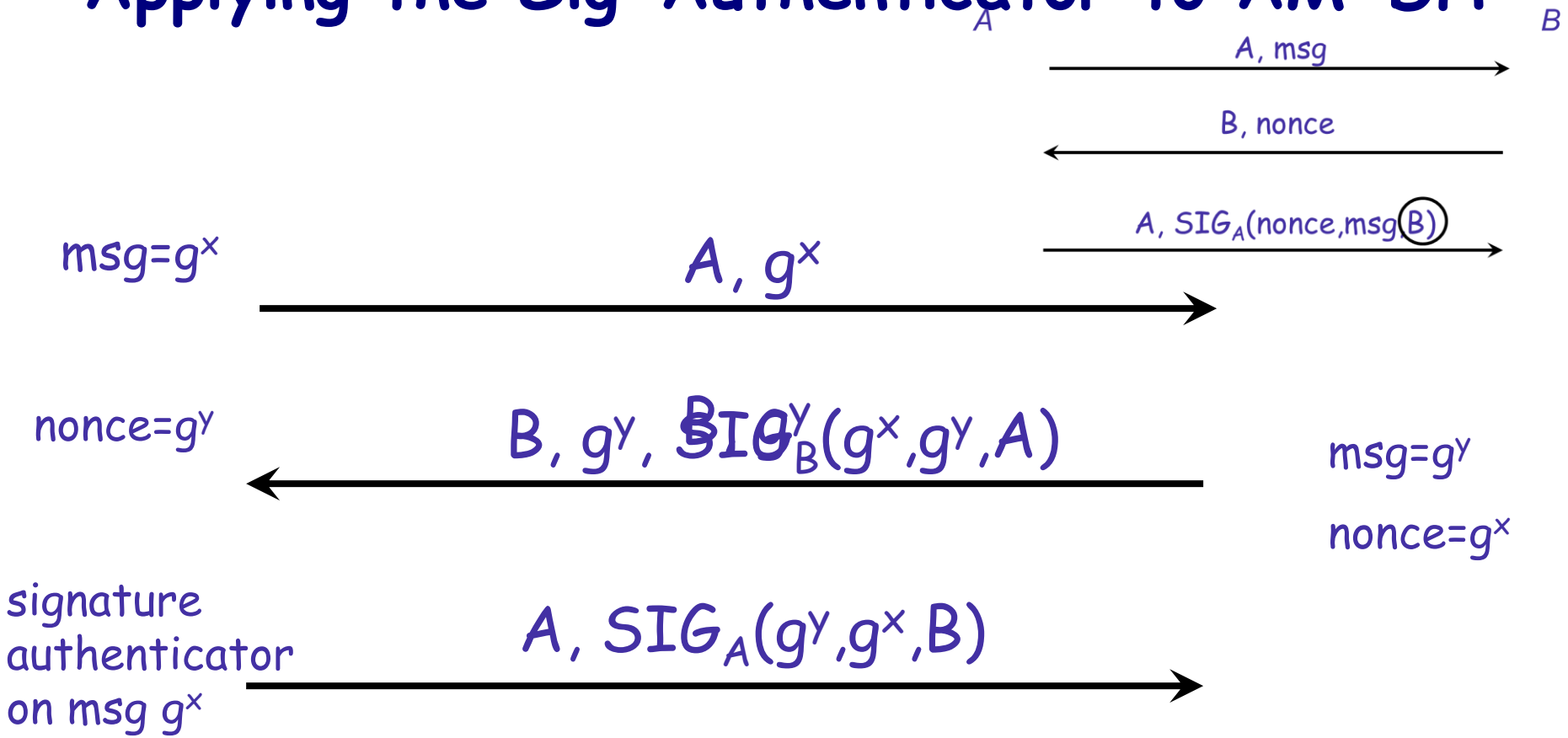
Proving ISO Using an Authenticator

- First prove basic DH is SK-secure in AM
  - Equivalent to Decisional DH assumption: $(g^x, g^y, g^{xy}) \approx_c (g^x, g^y, g^{random})$
    i.e., g^{xy} indistinguishable from random element in G



- Next apply the sig-based authenticator to this protocol ➔ a proof of the ISO protocol!!

Applying the Sig-Authenticator to AM-DH



We have ISO-AM-DH plus Signature-based authenticator

Authenticator applied to g^y is slightly different variant:
first A sends nonce (g^x), then B sends message (g^y) with signature

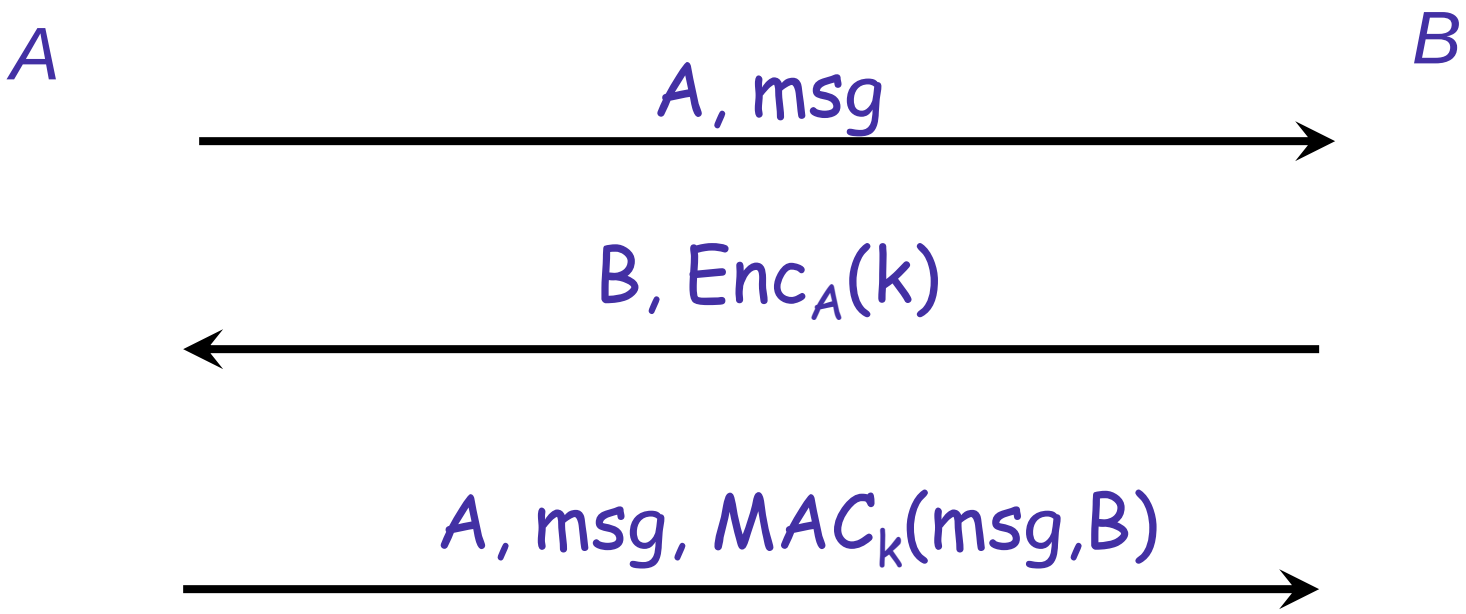
Conclusion: the ISO protocol is SK-secure
(QED: with a simple and intuitive proof)

Other Authenticators

(and the SKEME Protocol)

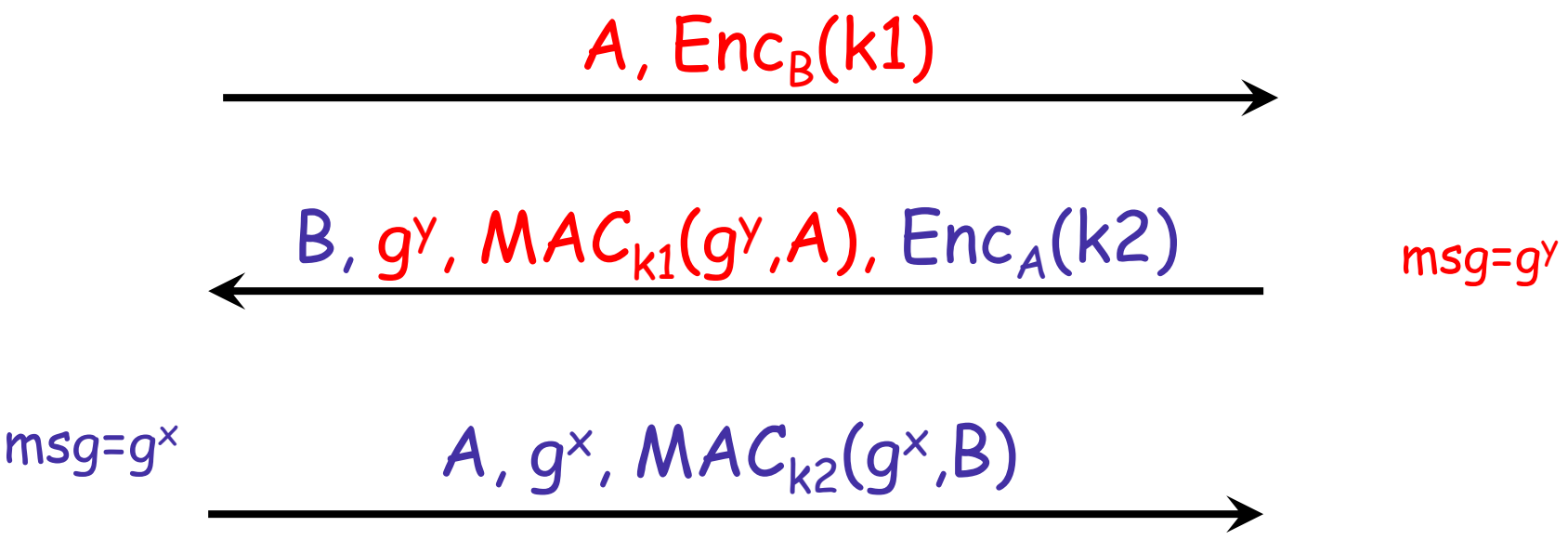
PK-Encryption-based authenticator

Single message authenticator: $A \xrightarrow{msg} B$:



Compiler from AM to UM: apply the above authenticator to each protocol's message

Applying the Enc-Authenticator to AM-DH



➔ the SKEME protocol [K'96, IKEv1]

Variants: •Key transport (no pfs)

. • Pre-shared key (with a MAC-based authenticator)

Authenticators are not always…

- Possible
  - Either the design is not decomposable into a basic AM-secure protocol and an authenticator applied to it
- Or desirable
  - The decomposition is artificial and adds more technicalities than understanding
- Yet when they "work" it usually results in a more intuitive, *modular* and easier-to-analyze protocol
  - And designing KE with authenticators in mind reduces the chances of hidden flaws

More on the Design of Key Exchange Protocols

- Privacy Issues: Identity Protection, deniability
- The design of the IKE Protocols: SKEME, SIGMA
  - "IPsec's Key Exchange" (IKEv1, IKEv2)

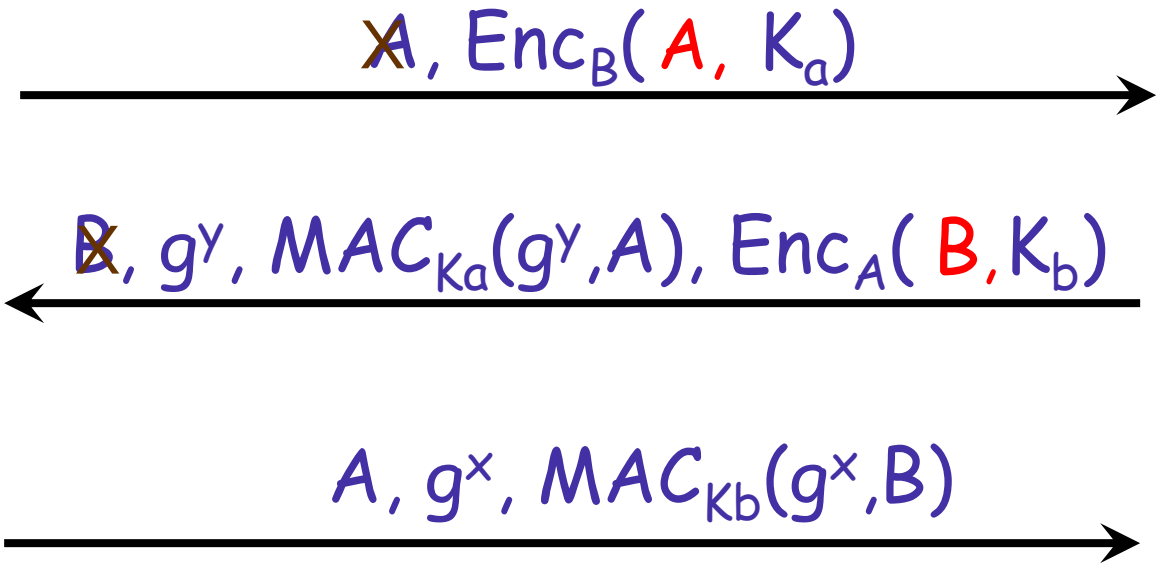
On Identity-Protecting KE Protocols

- Identity protection
  - Hiding identities from passive and/or active attackers
  - Logical identities (e.g. cert's) vs. physical addresses
- A privacy concern in many scenarios
  - Probing attacks in the Internet: who are you?
  - Location anonymity of roaming users
  - The "intelligent passport" application
- IPSec/IKE: design highly influenced by such privacy concerns (→ SKEME, SIGMA)

Identity Protection

- Passive vs. active attacker
  - Both id's protected against passive attacks but only one against active attacks
  - Which identity should get active defense?
    - Initiator: roaming user (e.g. hide location)
    - Responder: avoid probing attacks: who are you? (e.g. passport)
- Presents some design challenges: conflict between anonymity and authentication

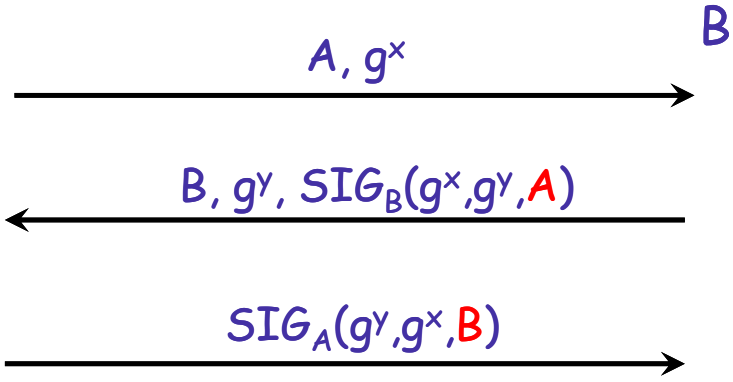
Identity Protection in SKEME



Issue: Id protection requires A to know B's pk (before run)

Next: SIGMA (signature based, solves this issue, IKEv2)

Why not ISO?



Unsuited for identity protection

- B needs to know A's identity before he can authenticate to A; same for A

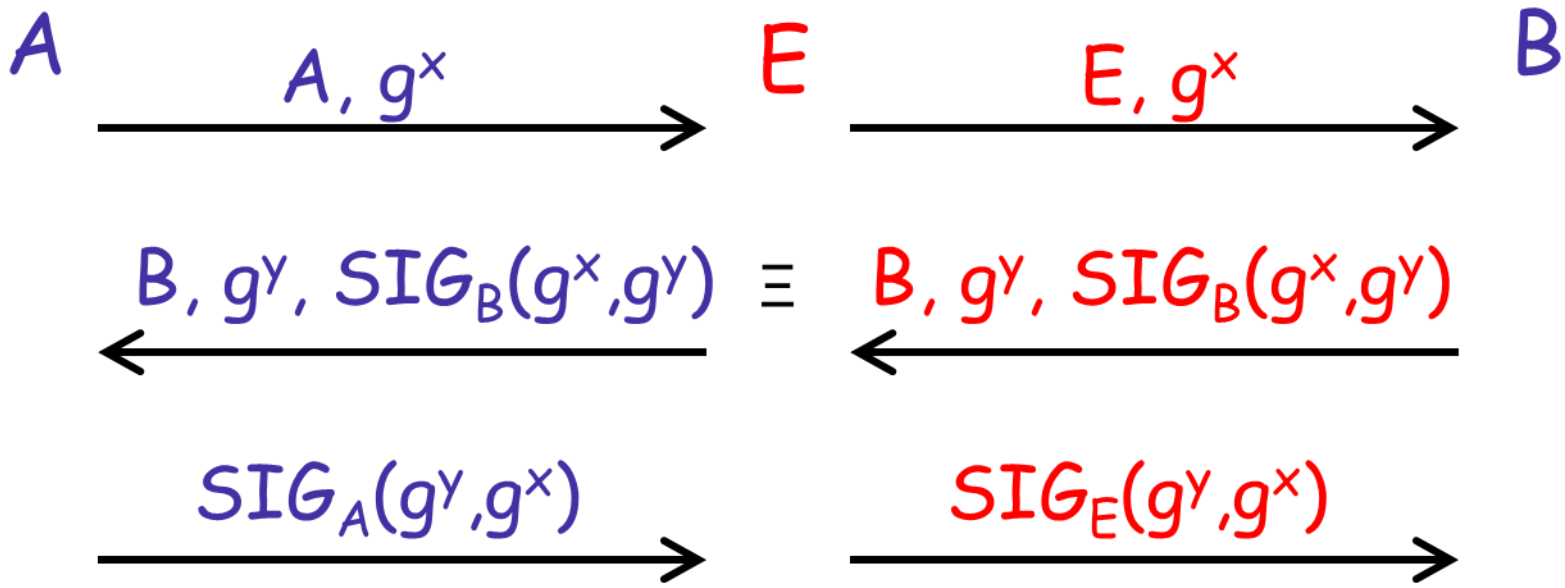
  ➔ Protection against active attackers is not possible

- Another privacy concern: leaving a signed proof of communication (signing the peer's identity)

- Letting each party sign its own identity rather than the peer's solves the privacy issues but makes the protocol insecure (the identity-misbinding attack again)

Alternative Solution: STS [DVW'92]

- Idea: to prevent the Id-M attack against BADH, A and B "prove knowledge" of $K=g^{xy}$ to each other
- Reminder Id-M attack (note that E doesn't know g^{xy})

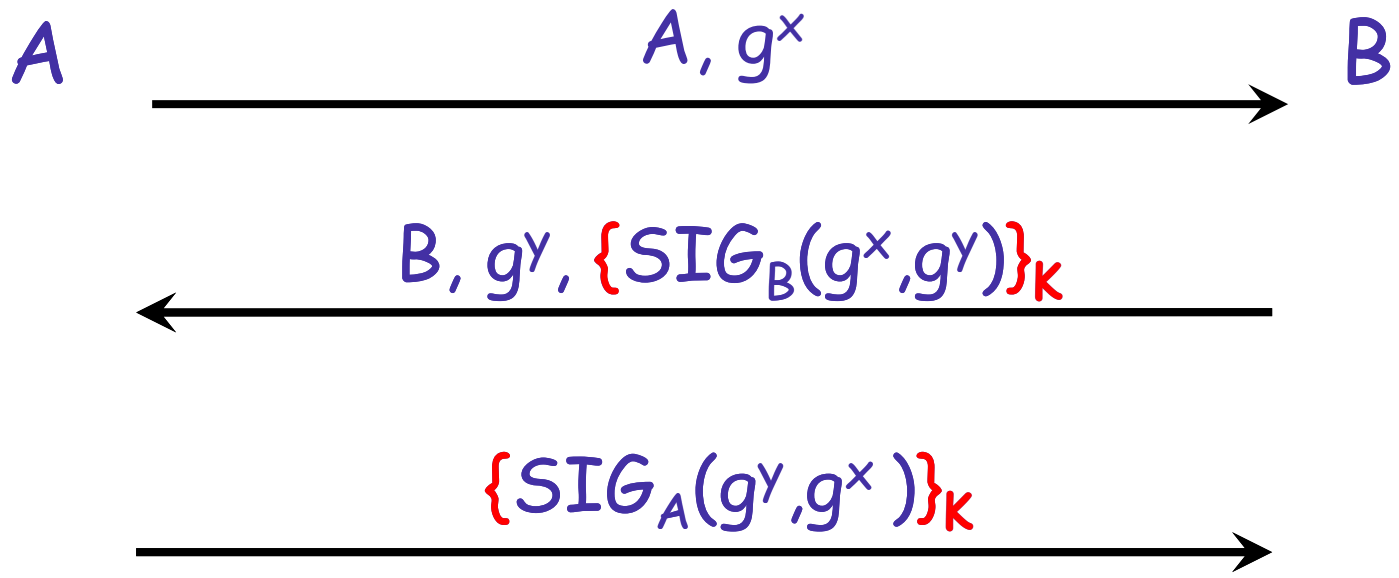


Alternative Solution: STS [DVW'92]

- Idea: each peer proves knowledge of $K=g^{xy}$

  (prevents the Id-M attack since in BADH E doesn't know g^{xy})

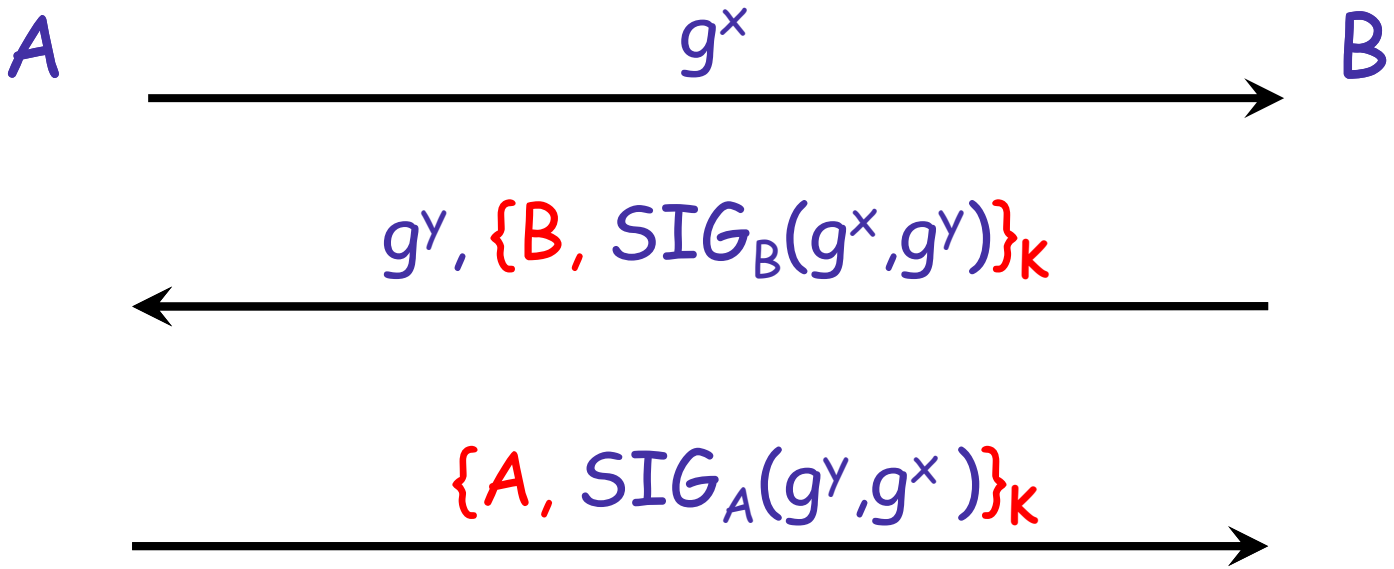
- As a "Proof of Knowledge" the STS protocol uses encryption under $K=g^{xy}$ (encryption denoted by $\{\ldots\}_K$)



STS Pro's and Con's

☺ Pro: STS can protect identities

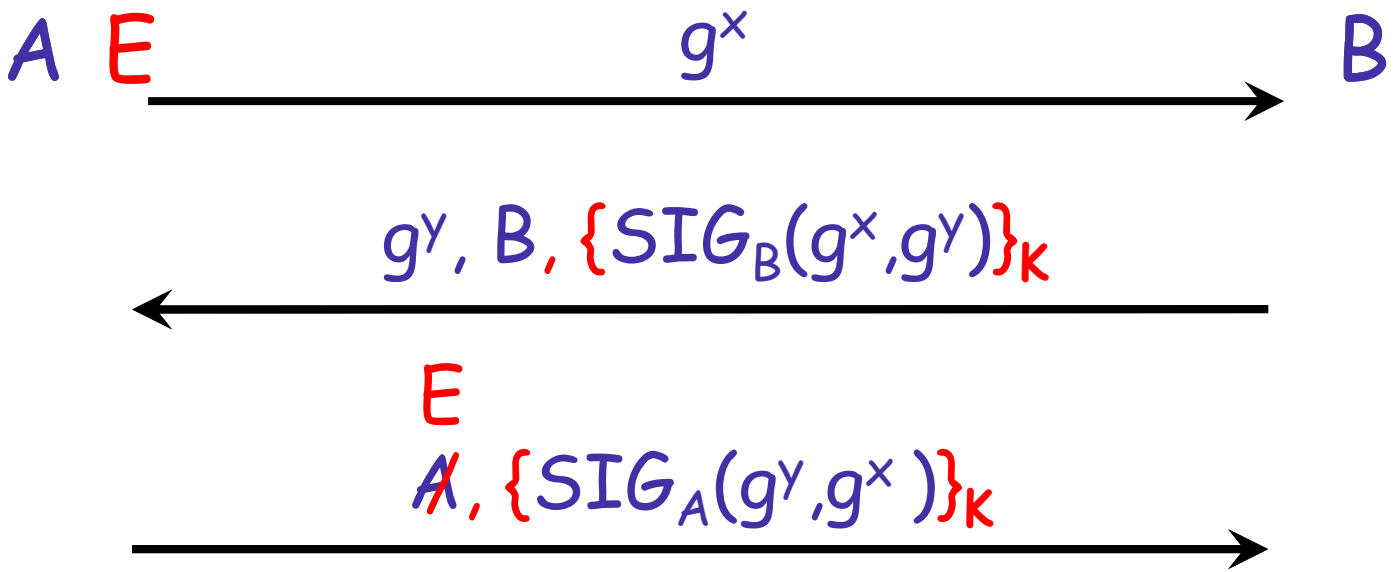
- Peer's id not needed for your own authentication
- Can extend encryption to cover identities (or cert's)



STS Pro's and Con's

☹ ☹ Con: Protocol is insecure! (encryption is not the right function to prove knowledge of a key)

- E.g.: if Eve can register A's public-key under her name she can mount the I-M attack (without knowing g^{xy})

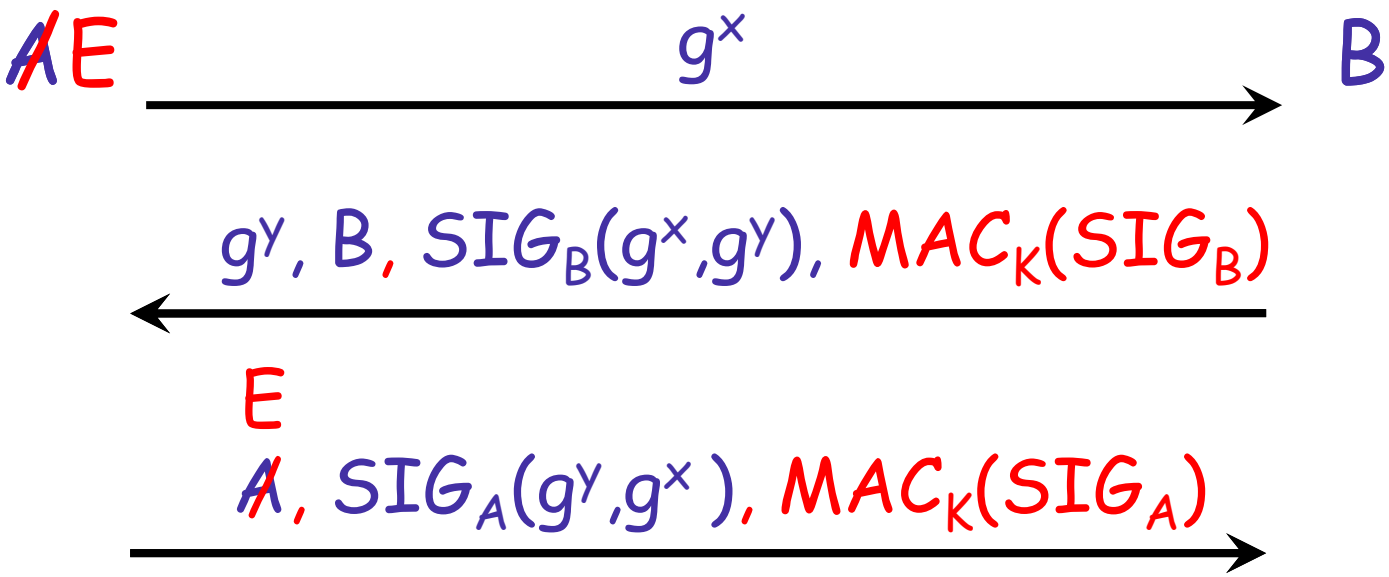


Identity-Misbinding on STS

- Eve registers A's PK as her own PK
  - Applicable when CA checks for identity of registrant but not for "possession" (PoP) of private key
- The attack is trivial if cert's not encrypted and trivial too if encrypted with a stream cipher
- Beyond the practicality of the attack, it is enough to show that "proof of knowledge of g^{xy}" via encryption is not enough.

MOREOVER…

STS with MAC (instead of encryption) [DVW]



- MAC_K better suited to provide Proof of Knowledge of K
- Yet: same attack as w/ encryption is possible!
- Can be mounted even if CA requires priv-key PoP! [BM99]
  - Even if signer's id put under sig ("on-line registration attack")

What is going on?

- The point is that "proof of knowledge" of $K=g^{xy}$ is not the issue
- What is required is:

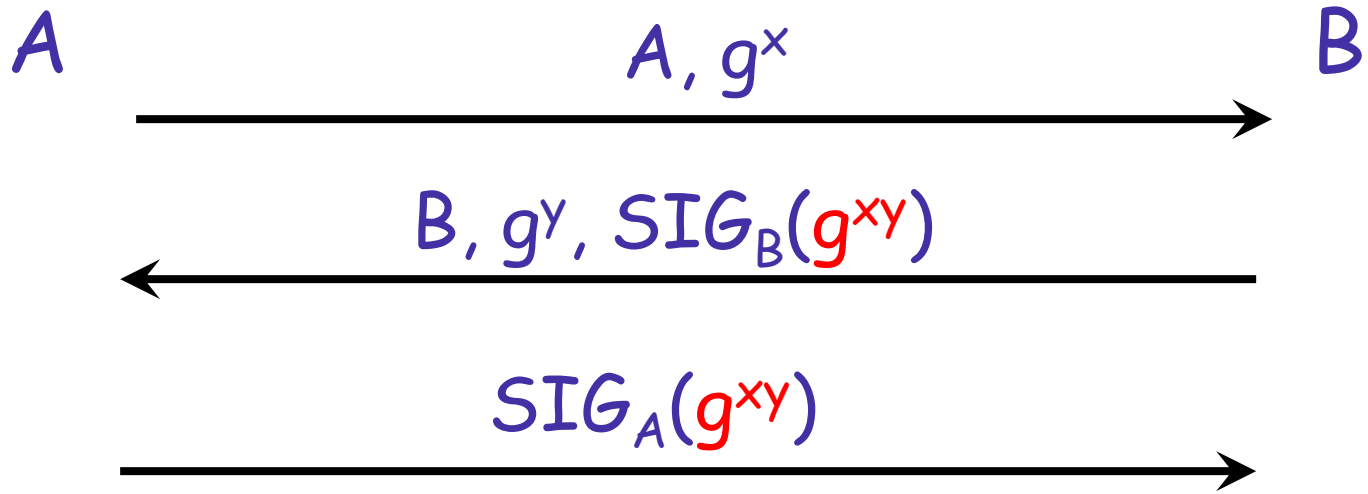
  binding the key K with the peer identities

- Which brings us to the SIGMA design
  - **SIG**n and **MA**c-your-own-identity!!
- And what about Photuris?
  - Yet another STS variant: Sign $K=g^{xy}$ as "proof of knowledge"; also insecure (see the SIGMA paper)
- But first another don't-do-it lesson: Photuris

But first another don't-do-it lesson: Photuris

Photuris Protocol (basic version)

Sign g^{xy} as direct authentication of g^{xy} and proof of knowledge of g^{xy}



- Id-Misb attack: Eve replaces $SIG_A(g^{xy})$ with $SIG_E(g^{xy})$, possible with RSA *(no need for E to register A's PK as her own)*

- SIG leaks information about g^{xy} e.g. $H(g^{xy})$ with RSA. *Breaks secure channel*
  if keys derived as HMAC(key=g^{xy}, data) which is computable given $H(g^{xy})$

- Small subgroup attack: next

Photuris Protocol (basic version)

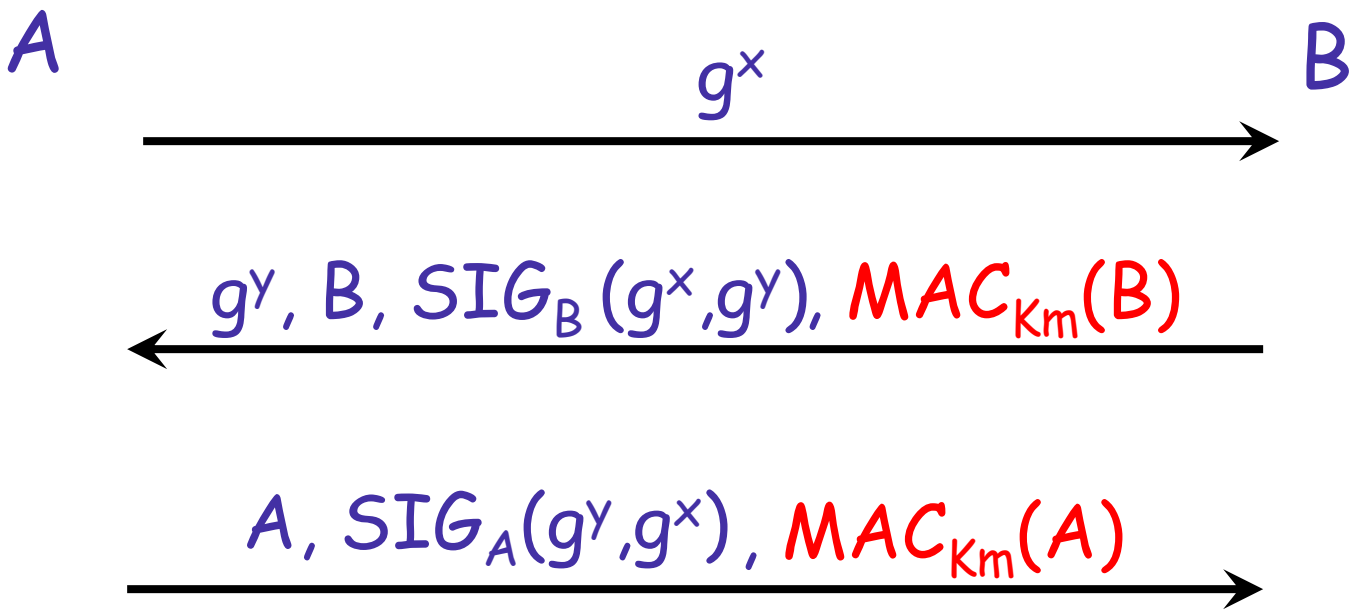
SMALL SUBGROUP ATTACK:

- Assume g in Z_p and p-1 has a small divisor s (i.e. $p=s \cdot t+1$, e.g. $s=3$)
- Attacker replaces g^x with $(g^x)^t$ and g^y with $(g^y)^t$
- A and B compute same key $K=g^{xyt}$
- But K now has order s, hence it only has s possible values! $(g^t, g^{2t},\ldots, g^{st})$

Adding g^x, g^y under sig solves the small group attack but not the other attacks

The SIGMA Protocols

SIGMA: Basic Version

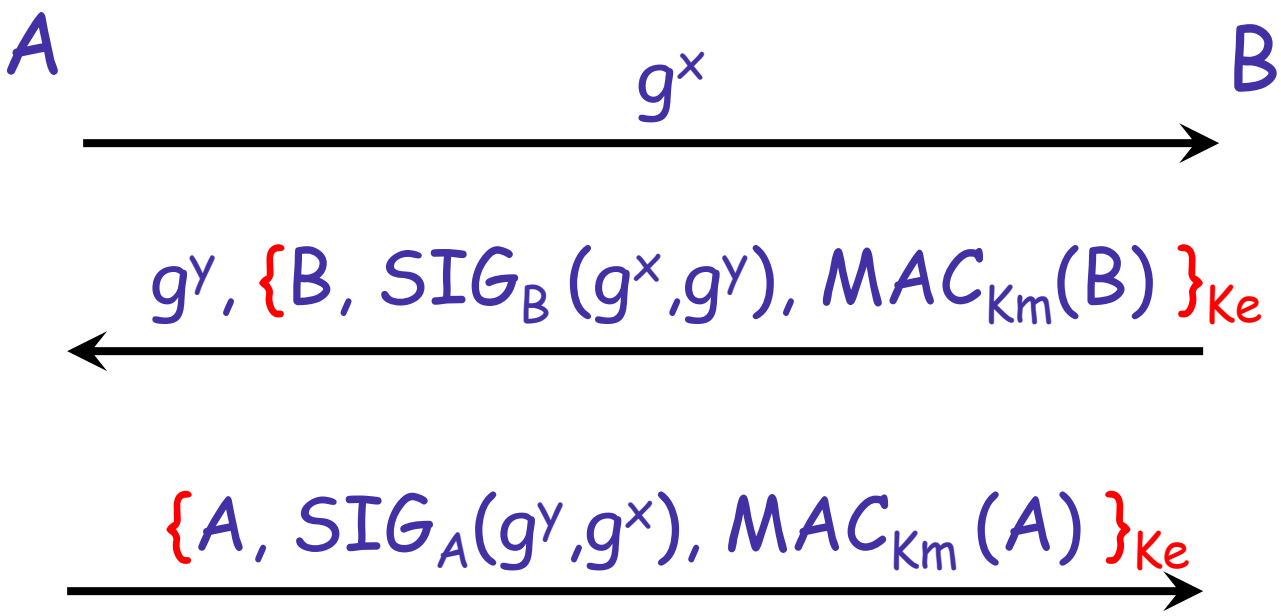


*K_m (and session key) derived from g^{xy} via a kdf

SIG and MAC: complementary roles (mitm and binding, resp)

Does not require knowing the peer's id for own authentication ➔ Great for id protection (& deniable)

SIGMA-I: active protection of Initiator's id

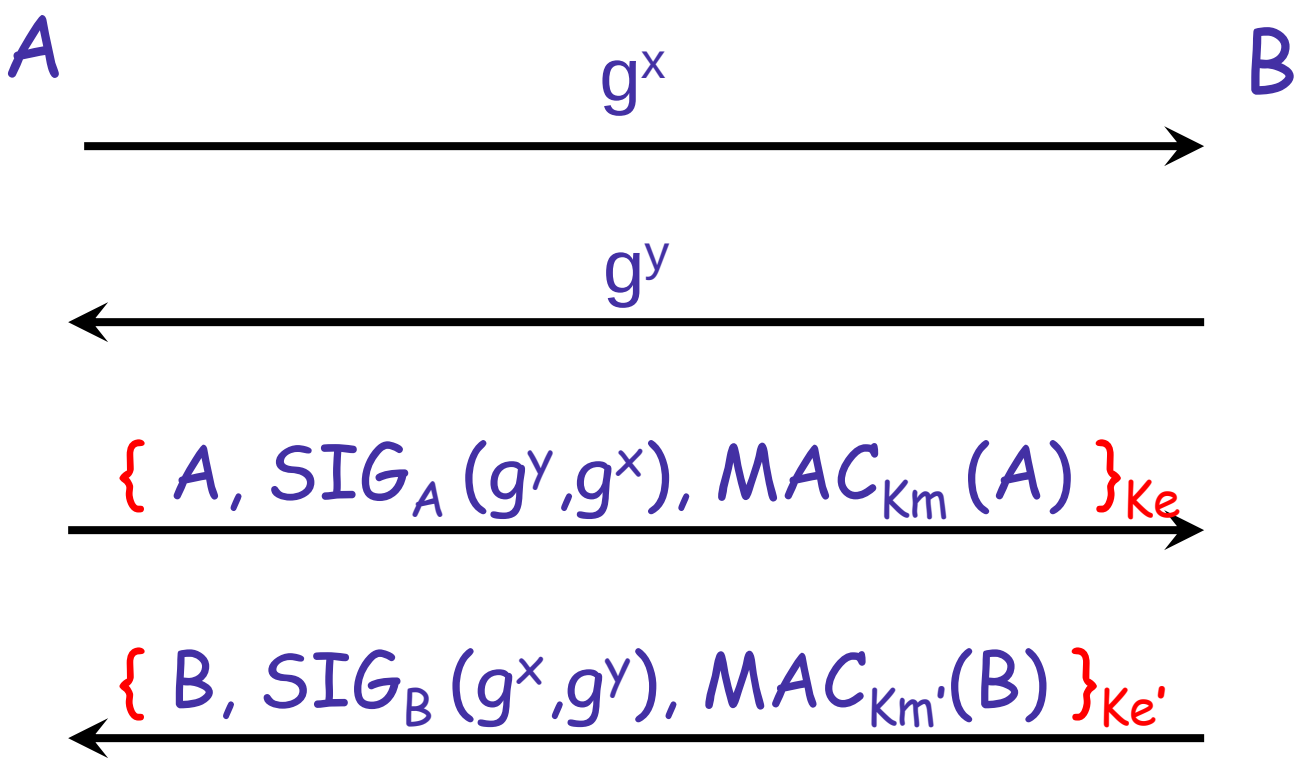


*K_e and K_m derived from g^{xy} via pseudorandom function

Responder (B) identifies first

➔ Initiator's (A) id protected

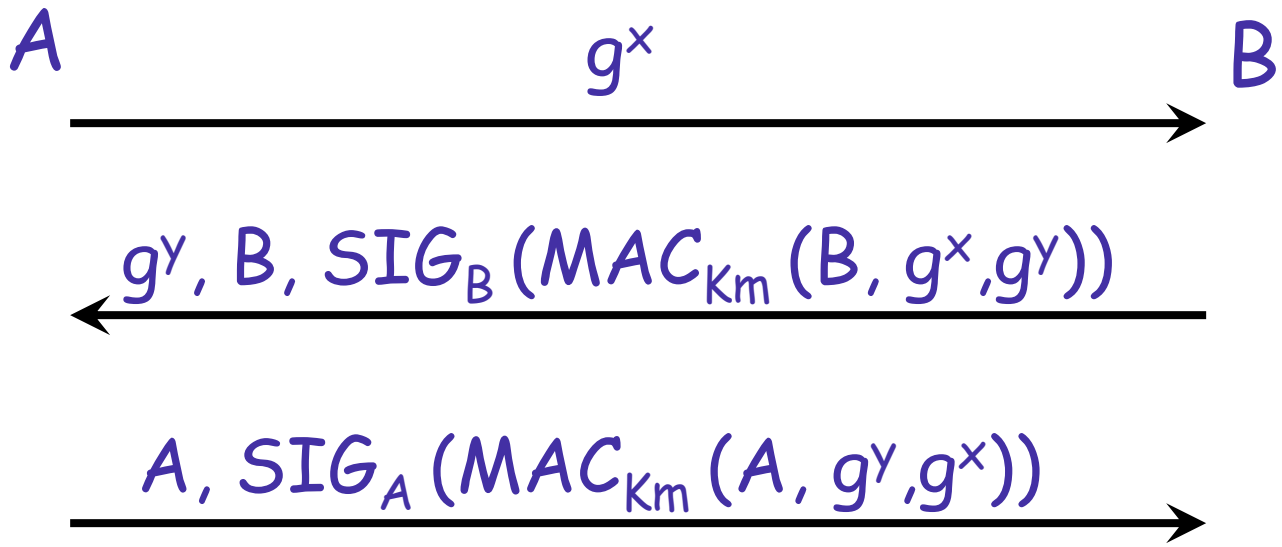
SIGMA-R: active protection of Responder's id



Note: Km, Km' and Ke, Ke' (against **reflection** attack)

IKEv1 Variant: MAC under SIG

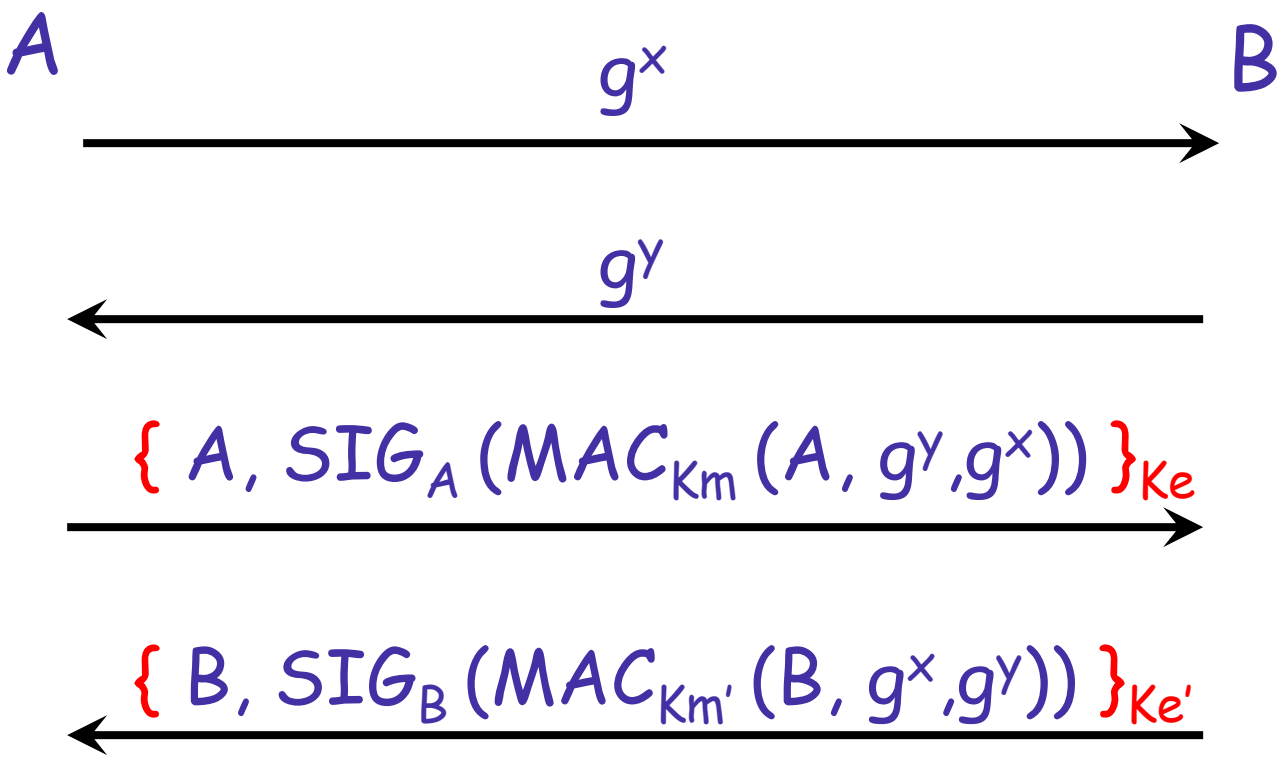
Equivalent security (just save MAC space):



➔ this is IKE's "aggressive mode" (no id protect'n)

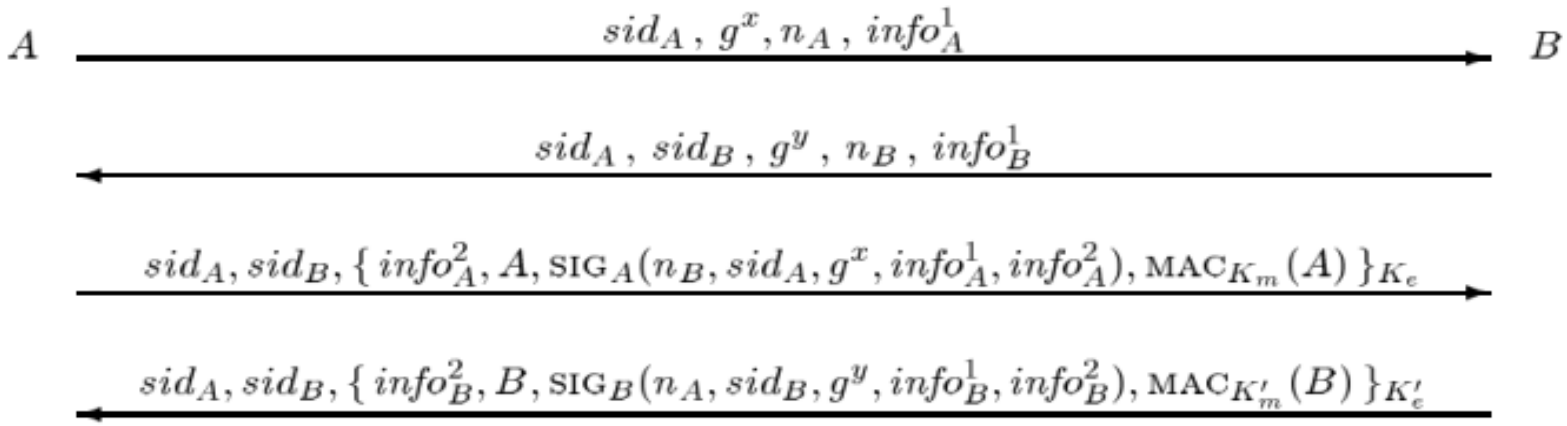
Note: MAC(SIG(id,...)) is not secure!! (STS-MAC)

IKE Main Mode



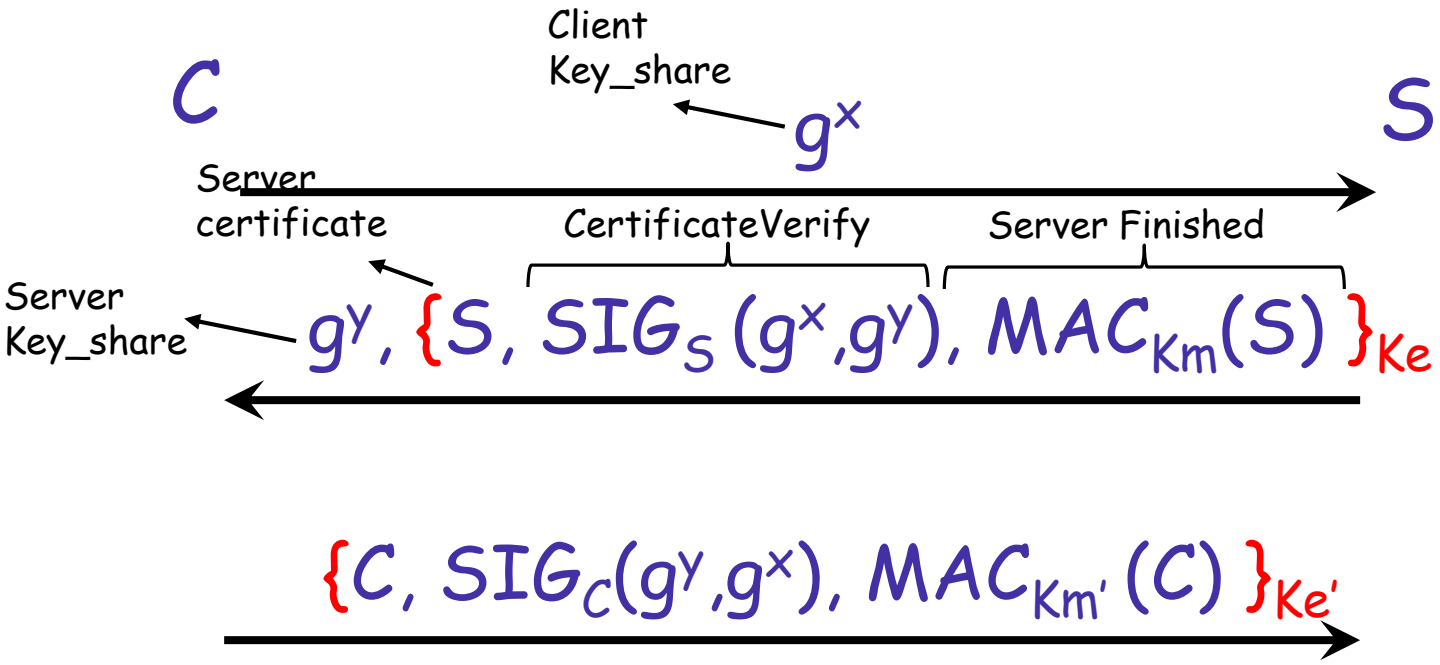
IKE v2: a slight variant – only MAC(id) under SIG

"Full fledge" SIGMA
(elements missing from skeleton figures)



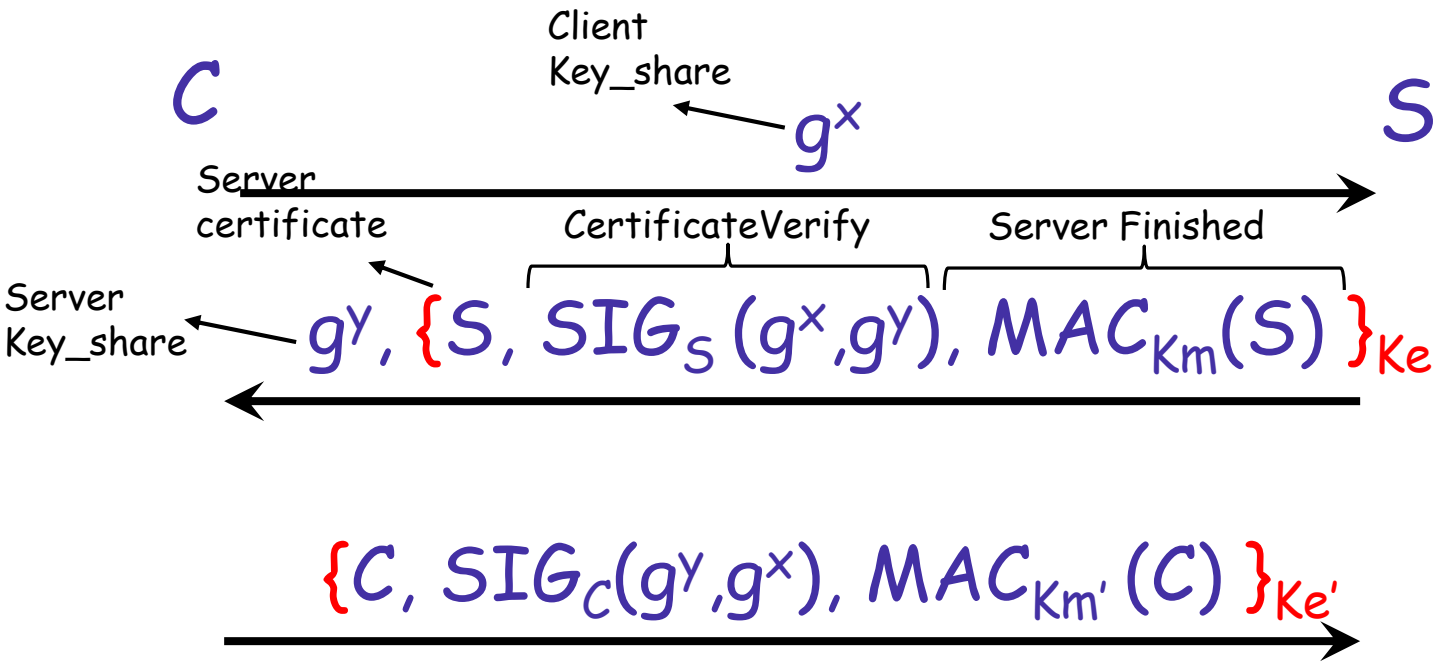
- Essential additional information (context, negotiation, etc.)
- Have separate elements for session identifiers, nonces, DH values
  - Basic principle: Don't use same element for multiple purposes
- Authenticate every element sent on the wire (identities too?)
  - Downgrade attacks

SIGMA-I ➔ TLS 1.3



*K_e , K_e' K_m , K_m' and session key derived from g^{xy} via KDF (handshake traffic / finished / application traffic keys)

SIGMA-I ➔ TLS 1.3



PLUS: hello msg: nonce + negotiation;

transcript-hash under signature and MAC

LESS: C, SIG_C for server-only authentication

SIGMA Summary

- SIGMA suitable for most applications requiring a Diffie-Hellman key exchange:
  - Simple and efficient (minimizes msgs and comput'n)
    - No over-design (nor under-design)
  - With or without ID Protection
  - Standardized: core key-exchange protocol for both IKEv1 and IKEv2, now also TLS 1.3
  - The "off-the-record communication" [Goldberg-Borisov] (use of deniability – proven in [DGK])

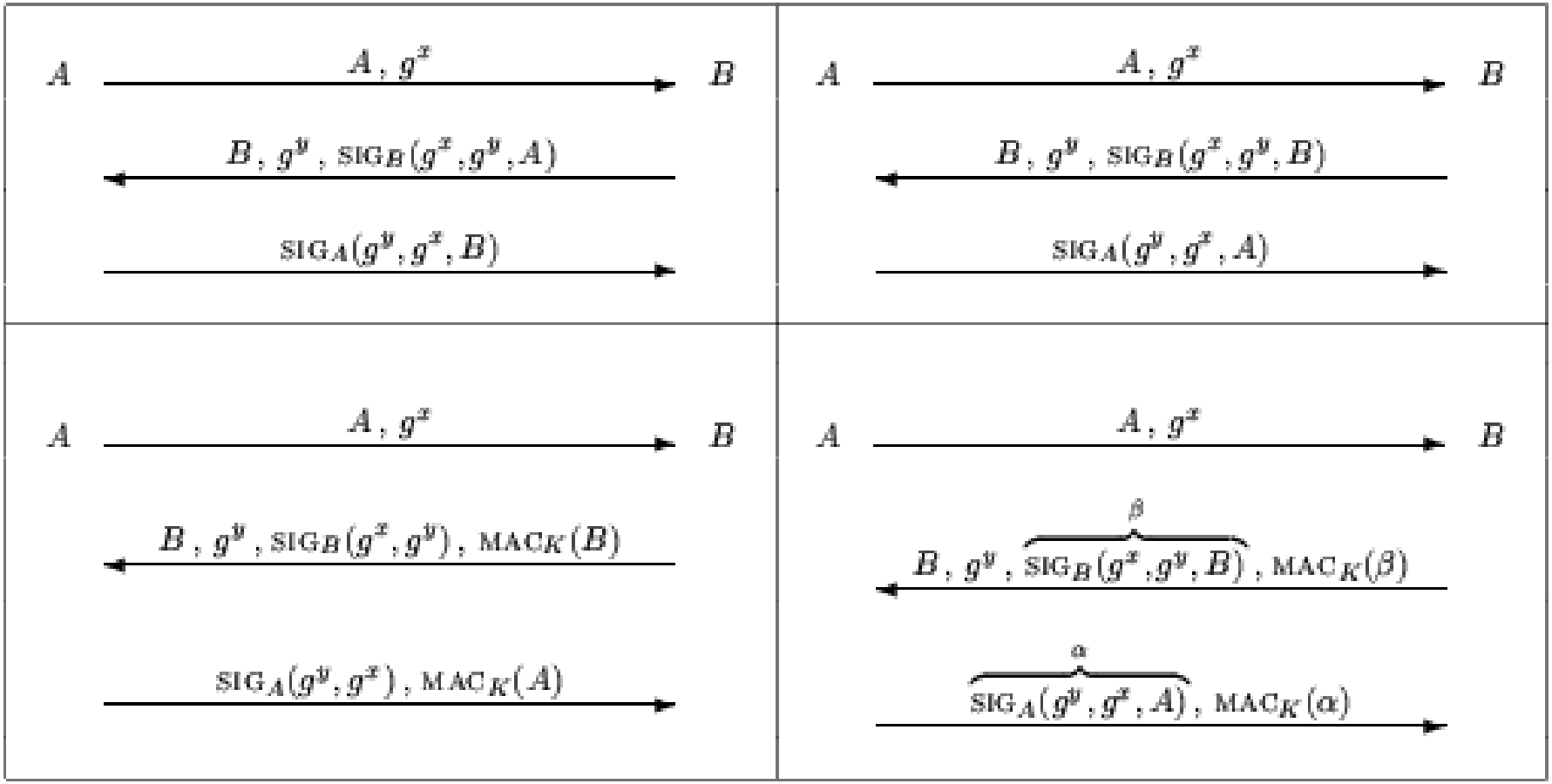
But is SIGMA Secure?

- Proof in Canetti-K Crypto'02
  - Formal proof: each element is essential
    - e.g., SIG(MAC(id,...)) vs. (SIG(id,...), MAC(SIG(id,...)))
  - Implies secure channels
  - Secure composition (universal composability, UC)

Care with variants!!

- From theory to practice
  - Specification, implementation, details, DoS, certificates, …

Care with Variants



KEM as Diffie-Hellman Generalization

- KEM: Key excapsulation mechanism ("key transport")
  - KEM(pk) → (c, K) De-KEM(sk, c) → K
- Generic DH:
  - Alice chooses pair (sk,pk) sends pk to Bob, keeps sk
  - Bob applies KEM(pk), sends c to Alice, outputs K
  - Alice compures K as De-KEM(sk, c), outputs K
- SKEME protocol: Long-term KEM + ephemeral KEM
  - Recent examples: OPTLS (TLS 1.3 proposal), Kyber (a lattice based "post-quantum" KE proposal), dispenses with signatures

Exercise

- Is this secure (as one-sided authentication)
- B→A: Nonce r
- A→B: $c=KEM(PK_B)$, $Sig_A(c,r)$
  - Session key $K=De\text{-}KEM(sk_B, c)$

Many more considerations…

- Negotiation: algorithm independ., downgrade protection
- Denial of Service protection
- Key derivation
- Secure channels, authenticated encryption
- Deniability
- …
- Automated analysis

PROOF-DRIVEN DESIGN®

- Proof-driven design: Formal analysis as main design tool
  - No simulation or empirical evidence; universal quantifier "for all"
- Proof guides choice of mechanisms, compose them right, discern between the essential, desirable and dispensable
- Result is efficiency, simplicity, rationale, even implementation guidance!
  - Simplicity as a security feature: Keep it simple! (but not simpler)