# Design and Analysis of Authenticated Diffie-Hellman Protocols

Hugo Krawczyk

IBM Research

hugo@ee.technion.ac.il



## Outline

- What is a Key Exchange Protocol
- Formalization and Design Challenges
- Authenticated Diffie-Hellman
- Lessons learned, Identity Misbinding Attack
- Formalizing and Proving Key Exchange Protocols
- Design: Authenticators and Modularity
- ISO and SKEME Protocols
- SIGMA Protocol and identity privacy
- When Modularity Fails: Arazi Protocol
- HMQV: Authentication almost for free

## Key Exchange

- Truly fundamental cryptographic protocol
  - The most (almost only) cryptographic protocol used in practice
    - I mean: protocol vs. function (encryption, signatures)
  - Two parties interact to agree on a common secret session key (from long-term to ephemeral keys)
  - Preamble to "secure channels"
    - E.g. SSL, IPsec (IKE), SSH
- Too many broken proposals
  - Few are secure, fewer have been proven

3

## Key Exchange Protocols

- 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: (x,K) ➔ x=A

4

## Key Exchange Protocols

- More generally:
  - n 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

5

## 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 reqts)
  - Ensure security of KE applications: "secure channels" as the quintessential application + composition
  - **Usability**: easy to analyze (stand alone → composable) + a design tool

6

## Designing and Analyzing KE Protocols…

- …is non-trivial
- Yet the end protocol need not be complex (only the way to get there may be)
  - And: to be practical the protocol *MUST BE SIMPLE*
- The best advice: learn from past experience (good and bad)
  - And remember: there is no ULTIMATE security model nor there are absolute proofs of security (but only relative to the model)

7

## First Part:

- Motivate security considerations for KE protocols through examples (and counter-examples)
  - Diffie-Hellman as the main example
- Sketch formalization of KE security [CK01,CK02]
- Some design and analysis methodology [BCK98] ("analysis as a design tool")
- Some covered protocols: ISO, SKEME, Arazi

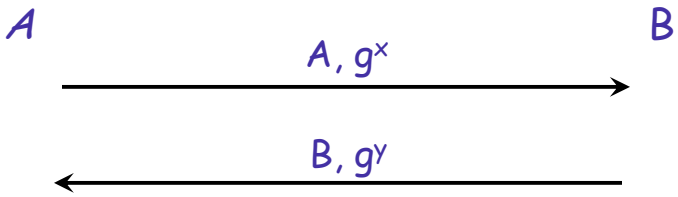
8

## 2nd Part

- Depending on time
- KE with ID Protection
  - The STS Protocol [DVW'92]
  - The SIGMA Protocol and IKE
- The MQV/HMQV Protocols:
  - efficiency vs proof complexity

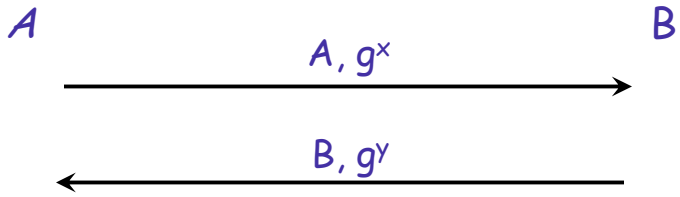


## Diffie-Hellman Exchange [DH'76]




- both parties compute the secret key $K=g^{xy}=(g^x)^y=(g^y)^x$
  - g = generator of a cyclic group G (e.g. $Z_p^*$, EC group)
- observer cannot distinguish K from uniform random in G
  - DDH 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)

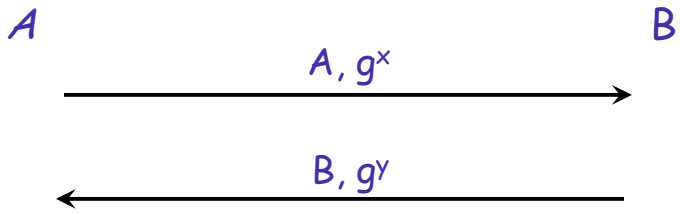
# Diffie-Hellman and PFS




- Maximal security: 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 gun point)
  - 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

11

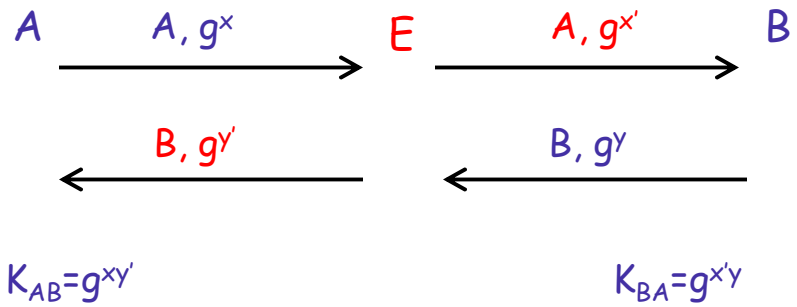
# Diffie-Hellman Exchange [DH'76]




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

12

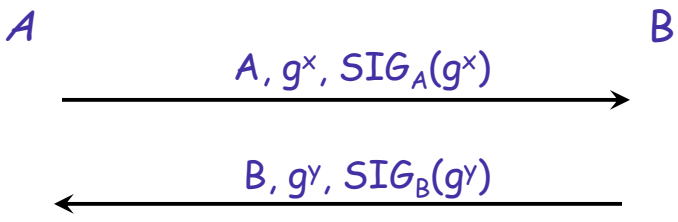
## (Wo)Man-in-the-Middle




Eve knows both keys!

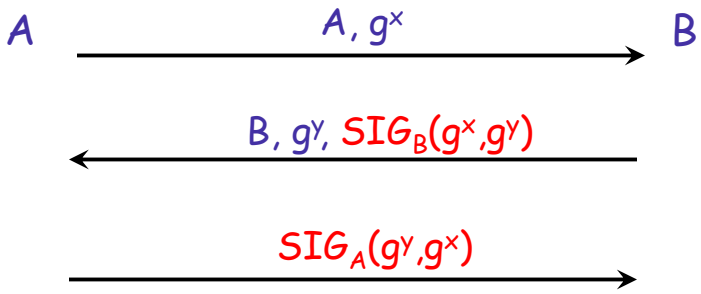


## Attempt at Authenticated DH




- what if attacker ever finds a triple $(x,g^x,SIG_A(g^x))$?
  - E.g., file of precomputed $(x,g^x)$ pairs
- Ephemeral leakage should not allow impersonation

# 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

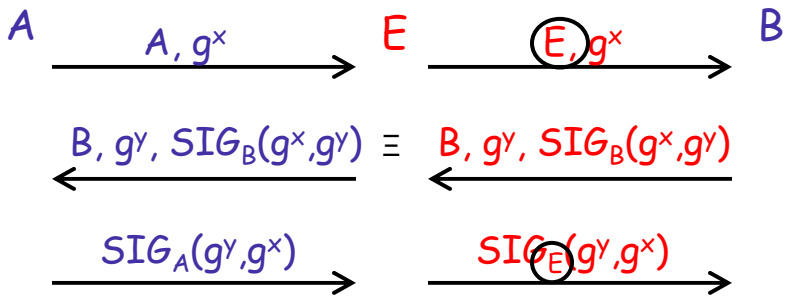
A: "Shared K=$g^{xy}$ with B" (K⇔B) B: "Shared K=$g^{xy}$ with A" (K⇔A)

**Looks fine, but...**

15

# 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⇔B) B: "Shared K=$g^{xy}$ with E" (K⇔E)

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

16

## 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$
- 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)

17

## Yet another example:

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

## B: "Shared K=$g^{xy}$ with E" (K⇔E)

- B=Central Command A=F-16 E= small unmanned plane
- B ⟶ E: {"destroy yourself"}$_K$
- E passes command {"destroy yourself"}$_K$ to A.
- Result: F-16 destroys itself!

18

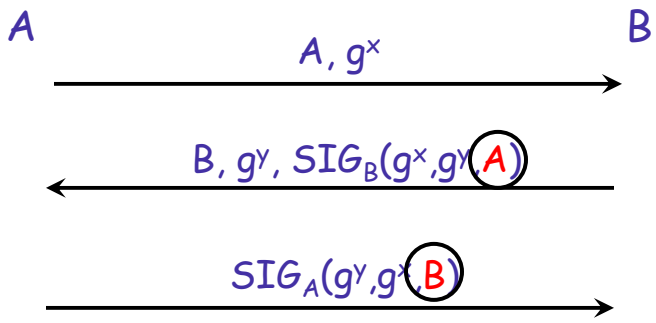
## Notes

- Attack discovered by Diffie-van Oorschot-Wiener [DVW'92]
  - the "differential cryptanalysis" of KE protocols
  - a reminder of the crucial consistency property
- The terminology Identity Misbinding Attack is mine

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

19

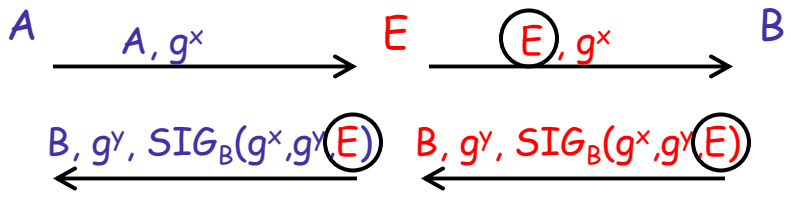
## A Possible Solution (ISO-9796)




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

20

## 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 above misbinding attack; but is it secure??

21

## The ISO Protocol is Secure

- We'll sketch the proof in the Canetti-Krawczyk (CK) model of key exchange [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 the CK model.

22

## 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
- Great recent "bridging" work towards "the best of two worlds" (e.g. Cortier, Fournet here)
- Here we focus on the first approach

23

## 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)

24

# Canetti-Krawczyk Model

- A combination of BCK'98 setting and BR'93 indistinguishability approach ("SK-security")
  - The goal: ensure good composition and modularity properties (as in BCK) but keep the simplicity of indistinguishability-based analysis ("usability")
- Secure channels as the must "test application"
  - Requires a formalization of secure channels (e.g., a transport protocol such as IPSec, SSL, SSH)
  - Definition of secure channels combines secure encryption and authentication against active attackers

25

# SK-Security: KE protocol

- A two-party protocol in a multi-party setting
- Many 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)

26

# SK-Security: Attacker

- Adversary model: unauthenticated links (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

27

# SK-Security: Attacker

- Adversary model: ~~u~~authenticated links (~~U~~AM)
  - Full control of communication links: monitors/controls/mo~~d~~ifies 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

28

## 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 a random value; it needs to guess which one is the case
   - We require that the probability that the attacker guesses right is not significantly better than a random guess (i.e. ½ + small ε)

29

## 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.
  - Known-key attacks: An important class of attacks studied separately: 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) 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)

30

## Note

- There is no guarantee for a session that is shared with a corrupted party.
- But there is a guarantee that interacting with the attacker does not compromise any session between honest parties
- Guarantee can be summarized as:
  - 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??

      Note: "key confirmation" possible but "common knowledge" is not
  - Another important property: unexposed session keys have "computationally independent" keys

31

## About public keys

- Parties have long-term secrets
  - e.g., private signature/decryption keys, or shared keys with other parties
- In the PK case: public keys of honest parties are 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 (such as Proof of Possession, checking structure of a key, etc.)

32

## SK-security results

- SK-security ➔ Secure Channels
  - Any key exchanged with an SK-secure KE protocol and used to "encrypt-then-authenticate" data realizes a secure channel [CK01]
- A variety of protocols have been proven SK-secure (both DH and key-transport): e.g., ISO, IKE (SKEME, SIGMA), HMQV, and more (also in the preshared-key model)
  - 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)

33

## SK-Security and Composition

- SK-Security preserved under authenticators
  - It suffices to prove a protocol secure in the ideally authenticated-links model (AM), and apply to it a simple "compiler" called *authenticator* (a design and analysis tool!)
  - We'll see an application to the proof of the ISO protoco
- CK02: SK-Security is "universally composable" (UC) (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)

34

## Authenticators [BCK98]

- Recall:
  - 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 DH in the AM is immediate (our first slide)

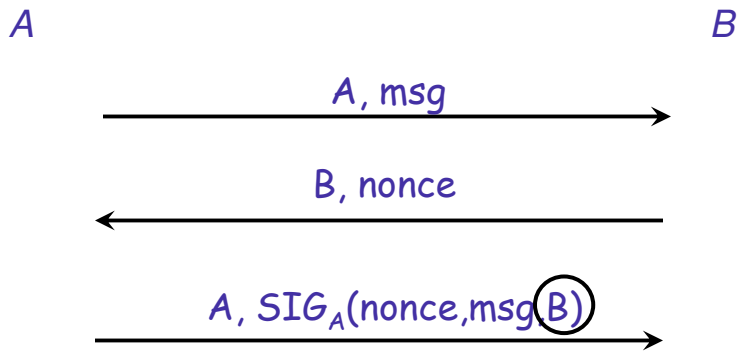
35

## 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 such that:
  - whenever A registers "received m from B", it also holds that B registered "sent m to B"
  - Note: To capture replay attacks: messages are assumed unique (e.g., concatenated with msg id or nonce)

36

## A signature-based authenticator

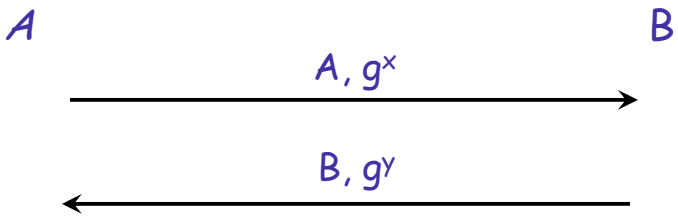



**Compiler from AM to UM:** apply the above authenticator to each protocol's message!!

37

## Proving ISO Using an Authenticator

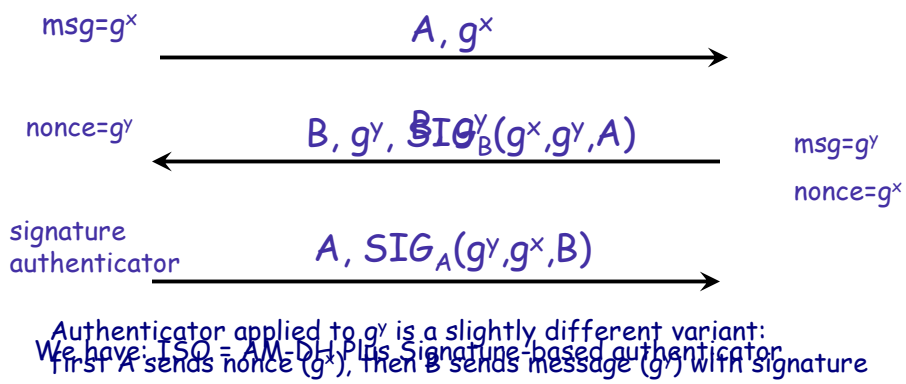
- First prove basic DH is SK-secure in AM (DDH assumption)




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

38

# Applying the Sig-Authenticator to AM-DH




We have: ISO = AM-DH Plus Signature-based authenticator

Authenticator applied to $g^y$ is a 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)

39

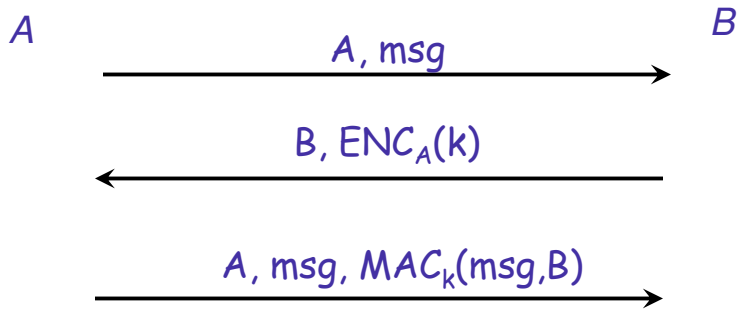
# Other Authenticators

(and the SKEME Protocol)

40

## Encryption-based authenticator

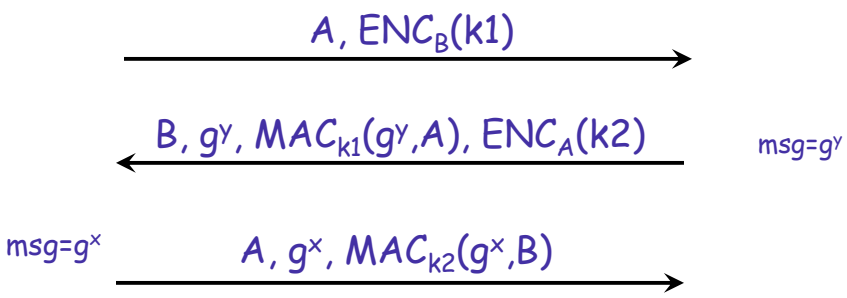
Single message authenticator: $A \xrightarrow{msg} B$:




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

41

## 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)

42

## 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 and easier-to-analyze protocol
  - And designing KE with authenticators in mind reduces the chances of hidden flaws

44

## More on the Design of Key Exchange Protocols

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

45

## 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

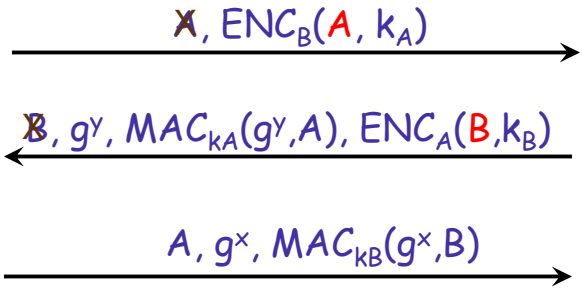
46

## Identity Protection

- Passive vs. active attacker
  - Best possible: both id's protected against passive attacks but only one against active attacks
  - Whose 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

47

# Identity Protection in SKEME

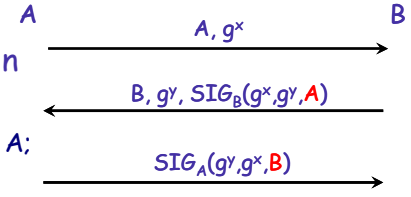



Issue: Id protection requires A to know B's pk (before run)

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

48

# Why not ISO?

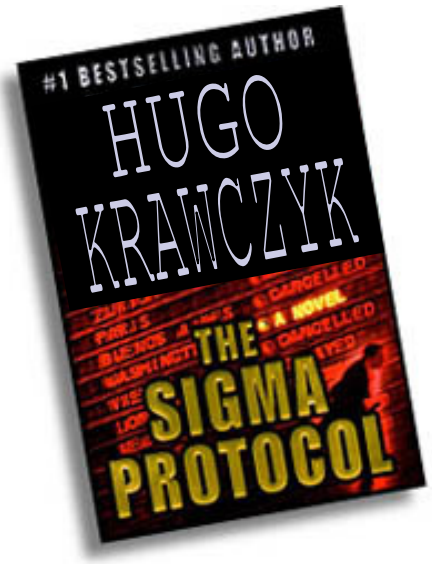



Unsuited for identity protection

- 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)
- An alias-based variant possible: see SIGMA paper

49

TI




50

## SIGMA: Basic Version

A → B: $g^x$

B → A: $g^y$, B, $SIG_B(g^x, g^y)$, $MAC_{Km}(B)$

A → B: A, $SIG_A(g^y, g^x)$, $MAC_{Km}(A)$

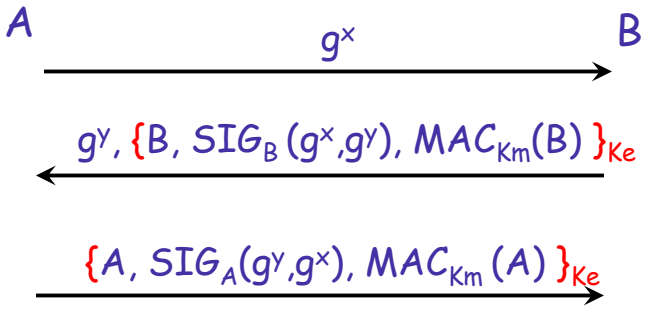
*$K_m$ (and session key) derived from $g^{xy}$ via a prg/prf

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)

51

## SIGMA-I: active protection of Initiator's id



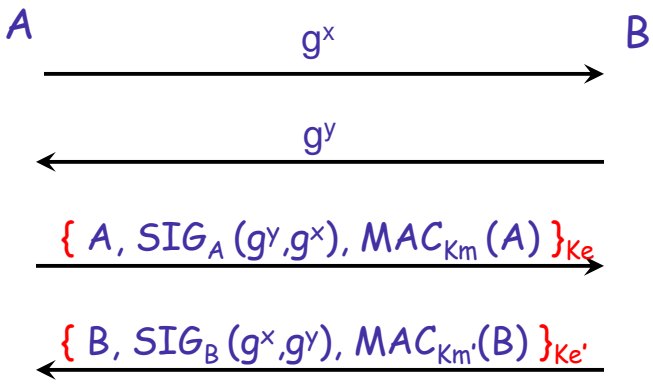

*$K_e$ and $K_m$ derived from $g^{xy}$ via pseudorandom function

Responder (B) identifies first

➔ Initiator's (A) id protected

52

## SIGMA-R: active protection of Responder's id




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

53

## IKEv1 Variant: MAC under SIG

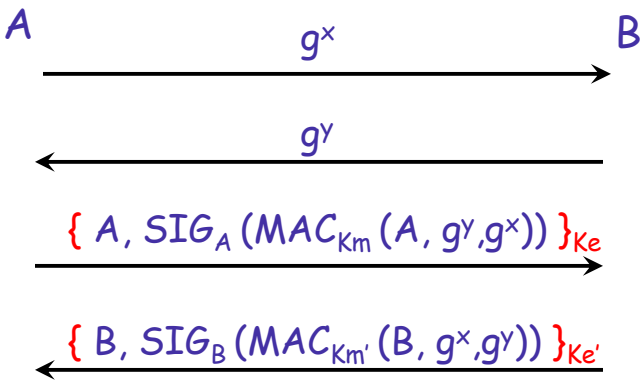
Equivalent security (just save MAC space):


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

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

54

## IKE Main Mode




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

55

## 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
    - Provides best possible protection (I or R protected against active attacks depending on application)
    - The "intelligent passport" application
  - Standardized: core key-exchange protocol for both IKEv1 and IKEv2

56

## But is SIGMA Secure?!

- **Secure!** (rigorous analysis): Canetti-K Crypto'02
  - Formal proof: each element is essential
    - e.g., SIG(MAC(id,...)) vs. (SIG(id,...), MAC(SIG(id,...)))
  - Guarantees secure channels
  - Secure composition with arbitrary applications (UC)

Care with variants!!

- From theory to practice
  - Specification, implementation, DETAILS (see "full fledge" appendix in paper -- web version)
  - DoS defenses: selective (IKEv2), integral (JFK-R)
  - ID Prot'n: Encryption secure against active attackers (RCCA)
  - Certificates, ...

57

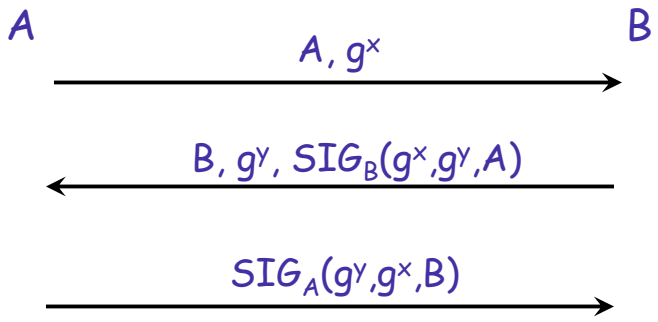
## A Revived (and Revised) Arazi Protocol



## Schnorr Signature

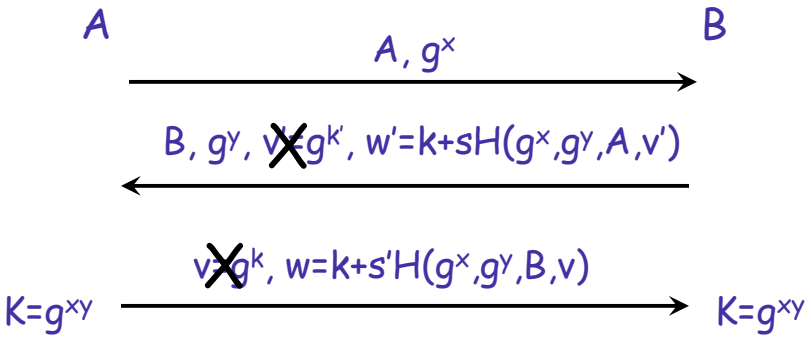
- Public parameters: Group G, generator g of prime order q, hash function $H:\{0,1\}^* \rightarrow Z_q$
- Private key s in $Z_q$ Public key $t=g^s$
- Signature on message M is pair (v,w)
  - $v = g^k$, k random in $Z_q$ (per signature value)
  - $w = k + s \cdot H(M,v) \mod q$
- Verification: check that $g^w = v \cdot t^{H(M,v)}$

## Recall ISO




60

## ISO w/Schnorr Signature




Natural question: Can we re-use the DH values $g^x$, $g^y$ as the v,v' in the protocol? (i.e. k=x, k'=y)

-- exponentiations are expensive!

61

## Security of the protocol

- Authenticators do not work anymore, but is there another proof?
- Nope. The protocol is insecure:

Set $h=H(g^x,g^y,A,v)$, $h'=H(g^x,g^y,B,v')$ then

$x=k=w-s\cdot h$ and $y=k'=w'-s\cdot h'$

$K = g^{xy} = g^{kk'} = g^{(w-s\cdot h)(w'-s'\cdot h')} = g^{ww'} \cdot t^{-hw'} \cdot t'^{-h'w} \cdot (g^{ss'})^{hh'}$

$g^{ss'} = (K/P)^{1/hh'}$ $\quad$ $=P$ (known value)

If a single session key K between A and B is ever learned then $g^{ss'}$ is learned and with it ALL A-B session keys!!

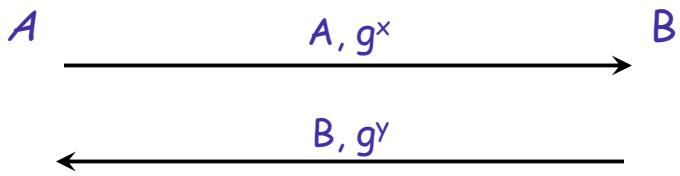
62

## Can Security be Saved?

- Yes. Set the session key to $K=H(g^{xy})$ where H is modeled as a "random oracle"
- Exercise 1: Prove it.
  Exercise 2: show the protocol does not have PFS.
- The above protocol is a derivative of Arazi's protocol'92 which used DSS instead of Schnorr (used only 2 msgs, did not hash the session key and used an "empty signature")
- The protocol then muted into other forms.
- Enter: "Implicitly authenticated DH"

63

## Implicitly Authenticated DH




- Authentication via session key computation
  - No transmitted signatures, MAC values, etc
  - Session key must involve long-term and ephemeral keys:
    $K=F(PK_A,PK_B,SK_A,SK_B,g^x,g^y,x,y)$
  - Ability to compute key ➔ authentication
- Possible and simple but tricky: many insecure proposals

64

## The HMQV Protocol:

**almost-free authentication!**

65

# The HMQV Protocol

- Basic DH + special key computation
- Notation: $G=\langle g\rangle$ of prime order $q$; $g$ in supergroup $G'$ (eg. EC, $Z^*_p$)
  - Alice's PK is $A=g^a$ and Bob's is $B=g^b$,
  - Exchanged ephemeral DH values are $X=g^x$, $Y=g^y$
- Set $d=H(X,\text{"Bob"})$ $e=H(Y,\text{"Alice"})$ (here H outputs $|q|/2$ bits)
- Both compute $\sigma=g^{(x+da)(y+eb)}$ as $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
- Session key $K=H(\sigma)$ (here H outputs $|K|$ bits, say 128)
- Authentication almost for free "½ exponentiation", no comm'n
- **Magic, isn't it?** Is it secure? Why? Can it be formally analyzed?

  Yes! A provable secure version of MQV (better performance too)

66

# HMQV Analysis

- HMQV: basic DH ($X=g^x$, $Y=g^y$), PKs: $A=g^a$, $B=g^b$
  - Compute $\sigma=g^{(x+da)(y+eb)}$ as $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
  - $d=H(X,\text{"Bob"})$ $e=H(Y,\text{"Alice"})$ (H outputs $\geq|q|/2$ bits)
  - Session key $K=H(\sigma)$ (e.g., 128 bits)
- No signatures exchanged, authentication achieved via computation of $\sigma$ (must ensure: only Alice and Bob can compute it)
- Idea: $(YB^e)^{x+da}$ is a sig of Alice on the pair (X, "Bob") and, at the same time, $(XA^d)^{y+eb}$ is a sig of Bob on (Y, "Alice")
  - Two signatures by two different parties (different priv/publ keys) on different msgs but with the same signature value!

67

## Underlying Primitive:
# Challenge-Response Signatures

- Bob is the signer (PK is $B=g^b$), Alice is the verifier (no PK)
  - Alice sends a "challenge" ($X=g^x$) and a msg m to Bob, who responds with a "challenge-specific" signature on m (sig depends on b, X, m)
  - Alice uses her "challenge trapdoor" (x) to verify the signature
- Alice→Bob: m, $X=g^x$

  Bob→Alice: $Y=g^y$, $\sigma=X^{y+eb}$ where $e=H(Y,m)$

  Alice accepts the signature as valid iff $(YB^e)^x=\sigma$
- Note: Alice could generate the signature by herself! (signature convinces only the challenger – non-transferable -- *bug or feature?*)
- We call this scheme XCR (Xponential Challenge Response)

68

# Security of XCR Signatures

- Theorem: XCR signatures are unforgeable
  - Unforgeability under usual adaptive chosen message attack
  - Only signer and challenger can compute it
  - Assumptions: Computational DH and H modeled as random oracle
- Idea of proof: "exponential" Schnorr via Fiat-Shamir (in a minute…)

69

## Dual XCR (DCR) Signatures

- Alice and Bob act as signers and verifiers simultaneously
- Alice has PK $A=g^a$, Bob has PK $B=g^b$
- Alice and Bob exchange values $X=g^x$, $Y=g^y$ and msgs $m_A$,$m_B$
- Bob generates an XCR sig on $m_A$ under challenge $XA^d$

  Alice generates an XCR sig on $m_B$ under challenge $YB^e$
- The signature is the same! $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
- This is exactly HMQV if one puts $m_A$="Alice", $m_B$="Bob" (since sig is the same value it needs not be transmitted!)

70

## Proof of HMQV

- Reduction from breaking HMQV as KE (in the CK model) to forging DCR
  - Not a trivial step
  - Great at showing the necessity of all elements in the protocol: drop any element and the proof shows you an attack (e.g. MQV)
- Reduction from forging DCR to forging XCR
  - Quite straightforward
- Reduction from forging XCR to solving CDH in RO model
  - I expand on this next

71

# XCR Proof via "Exponential Schnorr"

- Schnorr's protocol (given $B=g^b$, Bob proves knowledge of b)
  - Bob→Alice: $Y=g^y$
  - Alice→Bob: $e \in_R Z_q$
  - Bob→Alice: $s=eb+y$ (Alice checks $YB^e=g^s$)

  [FS]: ZK for honest verifier (Alice) ➔ $(Y,s=eb+y)$ w/ $e=H(m,Y)$ is a RO sig on m

- Exponential Schnorr: Bob proves *ability to compute* $()^b$
  - Bob→Alice: $Y=g^y\{0,1\}^{|q|/2}$
  - Alice→Bob: $e \in_R Z_q$, $X=g^x$
  - Bob→Alice: $\sigma=X^{eb+y}$ (Alice checks $(YB^e)^x=\sigma$)

  ZK for honest verifier (& any X) ➔ $(Y, \sigma=X^{eb+y})$ w/ $e=H(m,Y)$ is a RO XCR sig on m

  **Theorem: XCR is strongly CMA-unforgeable (CDH + RO)**



# Proof: A CDH solver C from XCR forger F

- Input: U, V in $G=\langle g\rangle$ (a CDH instance; goal: compute $g^{uv}$)
- Set $B = V$ $X_0 = U$ (B is signer's PK, $X_0$ is challenge to forger)
- Run F; for each msg m and challenge X queried by F (*a CMA attack*) *simulate* signature pair $(Y,X^s)$ (random s, e; $Y=g^s/B^e$; $H(Y,m) \leftarrow e$)
- When F outputs forgery $(Y_0, m_0, \sigma)$: (* $(Y_0,m_0)$ fresh and $H(Y_0,m_0)$ queried *)

  Re-run F with new independent oracle responses to $H(Y_0,m_0)$
- If 2nd run results in forgery $(Y_0, m_0, \sigma')$ (* same $(Y_0,m_0)$ as before! *) then C outputs $W=(\sigma/\sigma')^{1/c}$ where $c=(e-e')$ mod q .
  ($e, e'$ are the responses to $H(Y_0,m_0)$ in 1st and 2nd run, respectively)

Lemma: with non-negligible probability $W=DH(U,V)$

Proof: [PS] + $W= (\sigma/\sigma')^{1/c} = ( (Y_0B^e)^{x0} / (Y_0B^{e'})^{x0} )^{1/c} = ((B^c)^{x0})^{1/c} = B^{x0}$

## Implications for HMQV (* X → XA^d *)

- We used $W = (\sigma/\sigma')^{1/c} = ( (Y_0 B^e)^{x0} / (Y_0 B^{e'})^{x0} )^{1/c}$

  But can we divide by $Y_0 B^{e'}$? Yes if B and $Y_0$ in G (have inverses)

- B in G always true (chosen by honest signer) but what about $Y_0$ which is chosen by forger?
  - Do we need to check that $Y_0$ in G? (An extra exponentiation?)
  - No. If $G \subset R$, then enough to check $Y_0$ has inverse in R
    - E.g: $G = G_q = \langle g \rangle \subset Z_p^*$; $R = Z_p^*$; simply check Y in $Z_p$ and $Y \neq 0$

➔ HMQV needs no prime order verification! (later: only if exponent leak)

- Forger can query arbitrary msgs with arbitrary challenges X (even challenges not in group G) → No need for PoP or PK test in HMQV!

  (X becomes XAd and we do not need to check X nor A!)

➔ **Robust security of HMQV without extra complexity**

74

## Final Remark

- The KE area has matured to the point in which there is no reason to use unproven protocols
  - Addressing practicality does not require (or justify) giving up on rigorous analysis
  - Proofs not an absolute guarantee (relative to the security model), but the best available assurance
  - It is easy to design simple and secure key-exchange protocols, but it is easier to get them wrong…
- Proof-driven design: Formal analysis as main design tool
  - Guides us to choose secure mechanisms, compose them right, discern between the essential, desirable and dispensable
  - Result is efficiency, simplicity, rationale, even impl'n guidance!

75

# Cryptography as a Science!

- Intuition, ideas, cryptanalysis, new attacks…
  all necessary and important but:
- Formal analysis as main confidence tool
  - Not a Panacea: never stronger than the model it is based on
  - But well-defined mechanisms and properties: can be verified
    (not just "trust me, I have not been able to break it")
  - Even a cryptanalysis tool (e.g. UKS, LimLee attacks, KCI w/o hash,…)
- Formal analysis as main design tool
  - Guides us to choose secure mechanisms, compose them right,
    discern between the essential, desirable and dispensable
  - Result is efficiency, simplicity, rationale, even impl'n guidance!
- **Provable security: a strong weapon! (use with care!)**

76