Design and Analysis of Authenticated Diffie-Hellman Protocols

Dr. Qazi Ejaz Ali

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

$$A: (B,K) \text{ and } B: (x,K) \rightarrow x=A$$

Key Exchange Protocols

- More generally:
  - n parties; any two may exchange a key
  - Sessions: multiple simultaneous executions
- Adversary:
  - Monitors/controls/modifies traffic (m-i-t-m)
  - May corrupt parties: learns long-term secrets
  - May learn session-specific information: state/keys
- Security goal: preserve authenticity, secrecy and consistency of uncorrupted sessions

Formalizing Key Exchange

- An intuitive notion but hard to formalize
- Wish list:
  - 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) + a design tool

Designing and Analyzing KE Protocols…

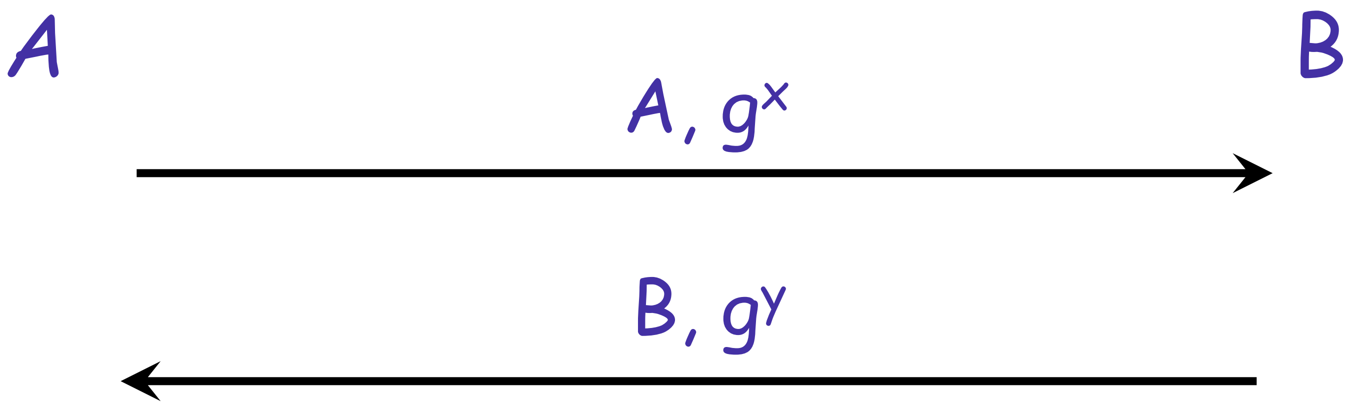
- …is non-trivial
- Yet the end product need not be complex (only the way to get there may be)
  - And: to be practical the protocol *MUST* be simple
- The best advice: learn from past experience (good and bad)
  - And remember: there is no ULTIMATE security model nor there are absolute proofs of security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

In this talk

- Motivate security considerations for KE protocols through examples (and counter-examples)
- Sketch formalization of KE security [CK01,CK02]
- Some design and analysis methodology [BCK98] ("analysis as a design tool")
- Diffie-Hellman as the main example
- Time permitting: KE with ID Protection
  - The SIGMA Protocol

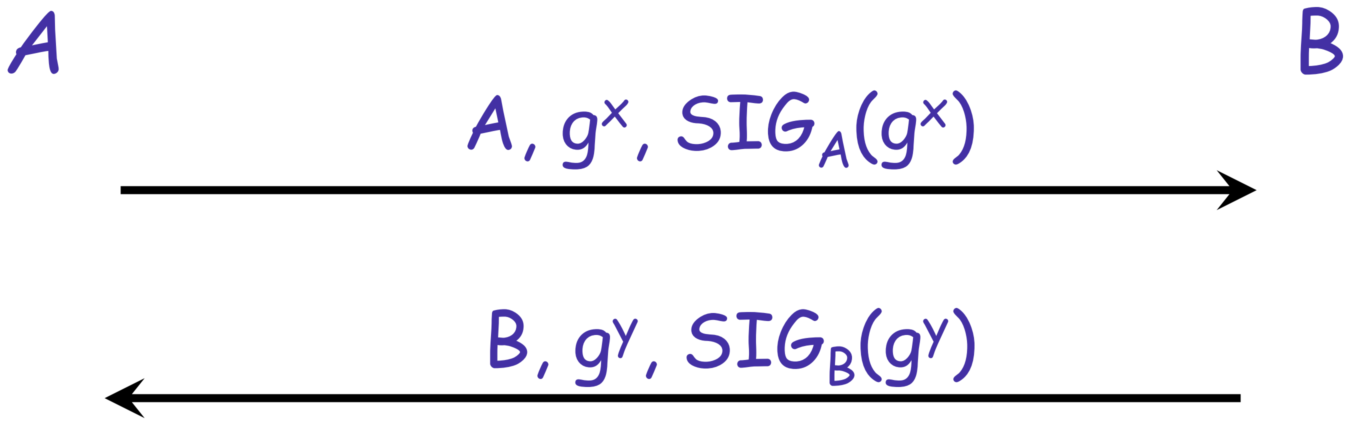
Example: Diffie-Hellman Exchange

- The original protocol [DH76]:



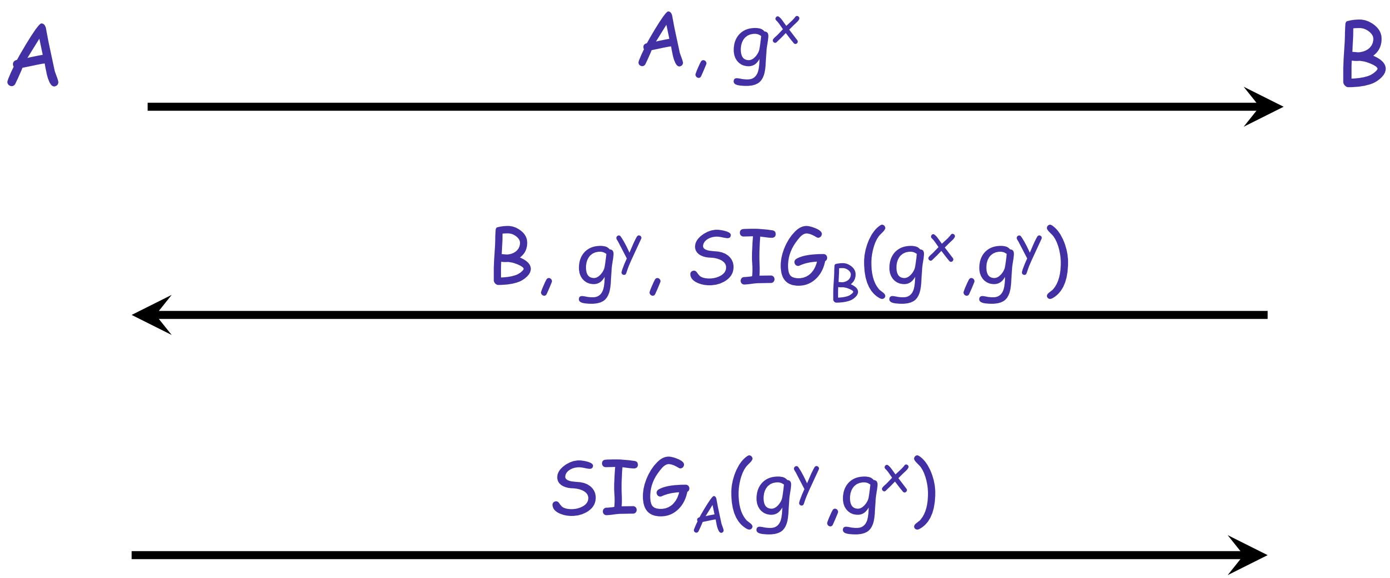
- both parties compute the secret key $K=g^{xy}$
- assumes authenticated channels (DDH assumption)
- open to m-i-t-m in a realistic unauthenticated setting

Authenticated Diffie-Hellman



- 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 never allow impersonation**

Basic Authenticated DH (BADH)



Peer's DH value acts as anti-replay nonce (I prefer explicit nonces)

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

Looks fine, but… **(there must be a reason we call it BADH)**

Identity-Misbinding Attack [DVW]

A → E: A, g^x | E → B: E, g^x

A ← E: B, g^y, $SIG_B(g^x,g^y)$ | E ← B: B, g^y, $SIG_B(g^x,g^y)$

A → E: $SIG_A(g^y,g^x)$ | E → B: $SIG_E(g^y,g^x)$

- 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 (e.g. $\{e\text{-cash}\}_K$)

Notes

- The above attack was discovered by Diffie-van Oorschot-Wiener [DVW'92]; it's the "differential cryptanalysis" of KE protocols – a reminder of the crucial consistency property
- The terminology Identity Misbinding Attack is from my "SIGMA" paper (Crypto'03)
- The attack is more commonly referred to as the Unknown Key-Share (UKS) attack.

A Possible Solution (ISO-9796)

A $\rightarrow$ B: A, g^x

A $\leftarrow$ B: $B, g^y, SIG_B(g^x, g^y, A)$

A $\rightarrow$ B: $SIG_A(g^y, g^x, B)$

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

The ISO defense

A → E: A, g^x

E → B: E, g^x

E ← B: B, g^y, $SIG_B(g^x, g^y, E)$

A ← E: B, g^y, $SIG_B(g^x, g^y, E)$

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 misbinding attack; but is it secure??

The ISO Protocol is Secure

- We will sketch the proof from Canetti-K (Euroc'01)
- Note that 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"

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
- Some recent “bridging” work
- Here we focus on the first approach
  - And in a small subset of works in the area

On KE Analysis: Bellare-Rogaway'93

- First complexity-theoretic treatment of KE
  - Indistinguishability approach [GM84]: attacker can't distinguish the real key from a random one
  - Authentication modeled via session "oracles"
- Prove several basic authentication and KE prot's (pre-shared secret key model)
  - Extended in [BJM97] to the PK-authenticat'n setting
- A subtle flaw (Rackoff): placing the distinguish test at the end of the run is insufficient

On KE Analysis: Bellare-Canetti-K'98

- Simulation-based definition of KE security
- Ideally-authenticated (AM) vs. real-life (UM)
- Modular authentication methodology
  - Authenticators: AM-to-UM compilers
- Goal: sec composition w/applications, sec channels
  - KE model too naïve: too strong, too weak (see Shoup'99)
  - A good tuning of the definition turned out to be tricky [CK02] (but the authentication techniques very useful!)

On KE Analysis: Canetti-K'01

- 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"
  - E.g., not achieved in original BR'93 formalization
  - Requires a formalization of secure channels (e.g., a transport protocol such as IPSec, SSL, SSH)
  - Definition of secure channels combines secure enc and auth against active attackers

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)
- Upon *completion* a session erases its state and outputs a triple: (session-id, peer-id, session-key)
- Sessions *named* by owner and session-id: e.g., (*A*,*s*)
  - CK01 uses the more technical notion of "matching sessions"; here we follow the simplified version presented in [SIGMA]; we assume "negotiated session-id's" (s_A,s_B)

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 or preshared master key)
  - May learn short-term information:
    - session state (e.g., the exponent x of a g^x value)
    - session key (of a present or past session)
- Terminology: *corrupted party, exposed session*

SK-Security Definition (simplified)

A KE protocol is *SK-secure* (in the above adversary model) if for any session (P,s) that completes at an uncorrupted party P with peer(P,s)=Q it holds:

1. If Q completes session (Q,s) while P and Q are uncorrupted then:
   a) peer(Q,s)=P; and
   b) sk(Q,s)=sk(P,s)
2. If sessions (P,s) and (Q,s) are not exposed, attacker cannot distinguish sk(P,s) from a random value

* this simplified formulation from [SIGMA] is slightly stronger than the one in [CK'01], cf. ENC protocol

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, SKEME, SIGMA, IKE, and pre-shared authenticated protocols
  - Two SK-secure flavors: with and w/o PFS
    (PFS modeled through session-expiration; expired sessions are NOT exposed even if attacker corrupts the session's owner)

SK-Security and Composition

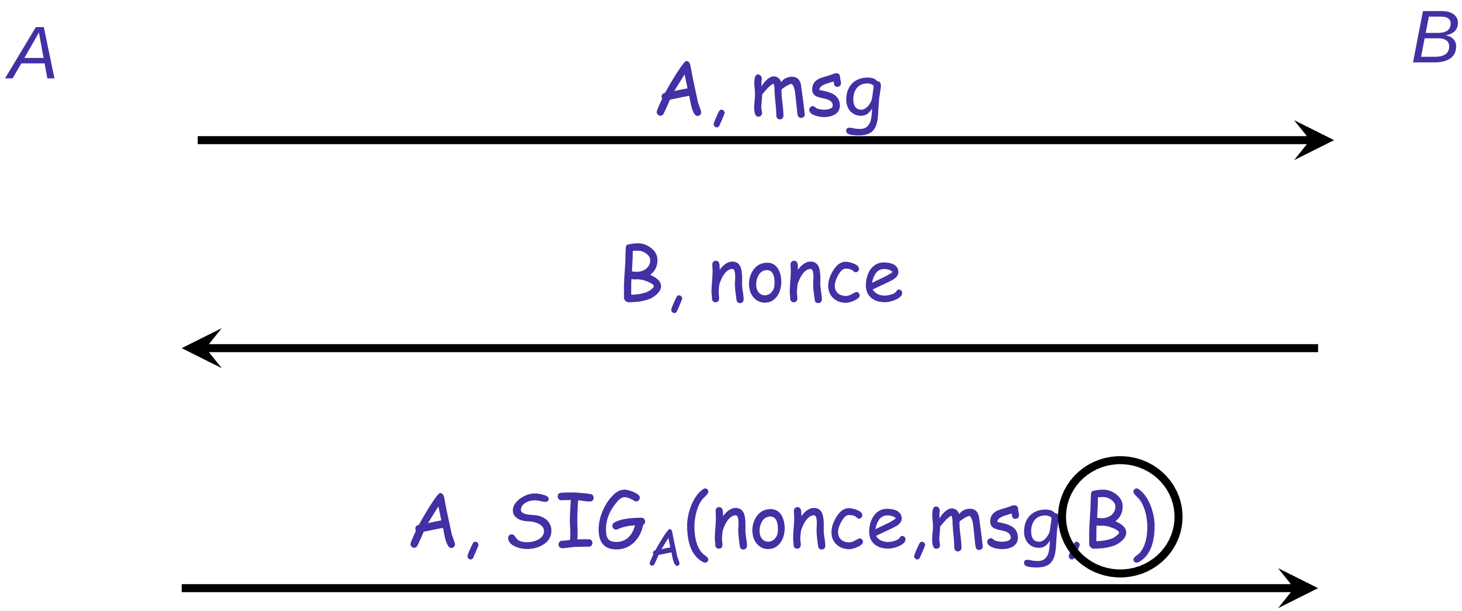
- SK-Security preserved under authenticators
  - It suffices to prove a protocol secure in the ideally authenticated-links model (AM), and apply to it an authenticator (both a design and analysis tool)
  - We'll see an application to the proof of the ISO prot'l
- CK02: SK-Security is "universally composable" (UC) (remains secure under composition with any application – not just secure channels)
  - Well, almost: true for protocols with the ACK property
  - True always if we weaken UC security via "non-information oracles" (see CK02 eprint/2002/059)

Authenticators [BCK98]

- Recall:
  - UM (Unauthenticated-links Model):
    a realistic attack model as described before
  - AM (ideally Authenticated-links Model): like UM but attacker cannot change or inject messages to 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

A signature-based authenticator

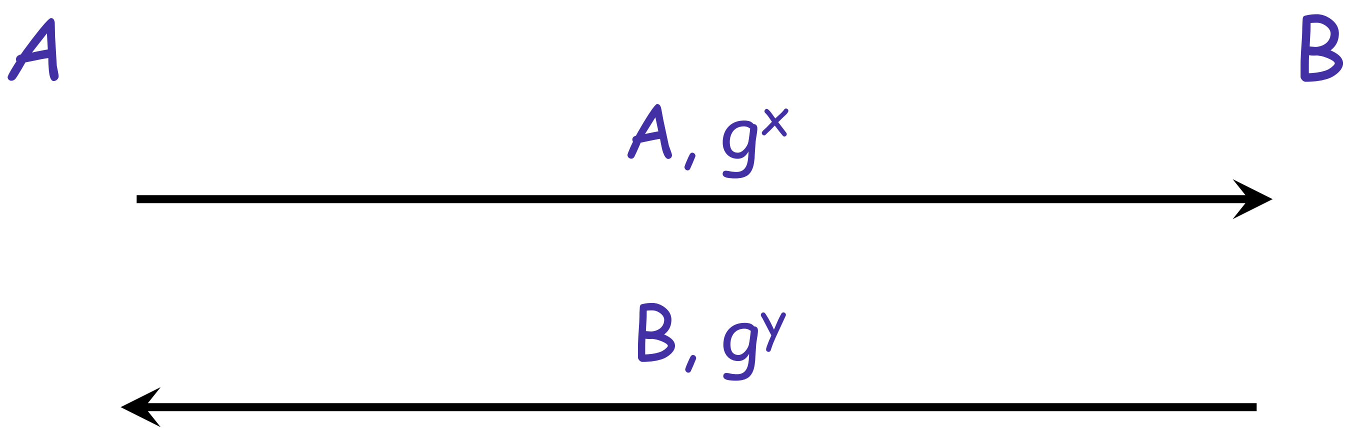
Single message authenticator: $A \xrightarrow{msg} B$:



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

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!!

Applying the Sig-Authenticator to AM-DH

msg=g^x — A, g^x →

nonce=g^y — ← $B, g^y, SIG_B(g^x, g^y, A)$ — msg=g^y

nonce=g^x

$A, SIG_A(g^y, g^x, B)$ →

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 (with a simple and intuitive proof)

Other Authenticators

- PK Encryption Based: applied to DH gives proof of main SKEME mode
  - Applied to a key-transport protocol provides a proof of the non-pfs mode of SKEME
- Pre-shared Key Authenticator: used to prove a simple re-key protocol and DH authenticated with a pre-shared key
- Note: different combinations of AM-secure protocols w/ different authenticators
  - In particular: public-key and shared-key mechanisms

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
    - maybe even the risk of heart attack… ☺

Conclusions

- Design of KE protocols is a subtle matter, formalizing their security too
- The AM-to-UM methodology via authenticators -- attractive: design and analysis
- SK-security: the convenience of indistinguisha-bility, the power of simulatability
  - In particular: secure channels and composition
  - Many practical KE protocols analyzed (esp auth'd DH): ISO, SKEME, SIGMA (last two: id protection), IKE
    - Symmetric and asymmetric authentication

Final Conclusion

- The KE area has matured to the point in which there is no reason to use unproven protocols
  - Do not leave home without a proof…

ThAnKs !!