

CSE 3400 - Introduction to Computer & Network Security
(aka: Introduction to Cybersecurity)

Lecture 9

Shared Key Protocols – Part II

Ghada Almashaqbeh

UConn

From Textbook Slides by Prof. Amir Herzberg

UConn

Outline

- Handshake protocol extensions.
- Key distribution centers.
- Improving resilience to key exposure.

Handshake Protocols Extensions

Authenticated Request-Response Protocols

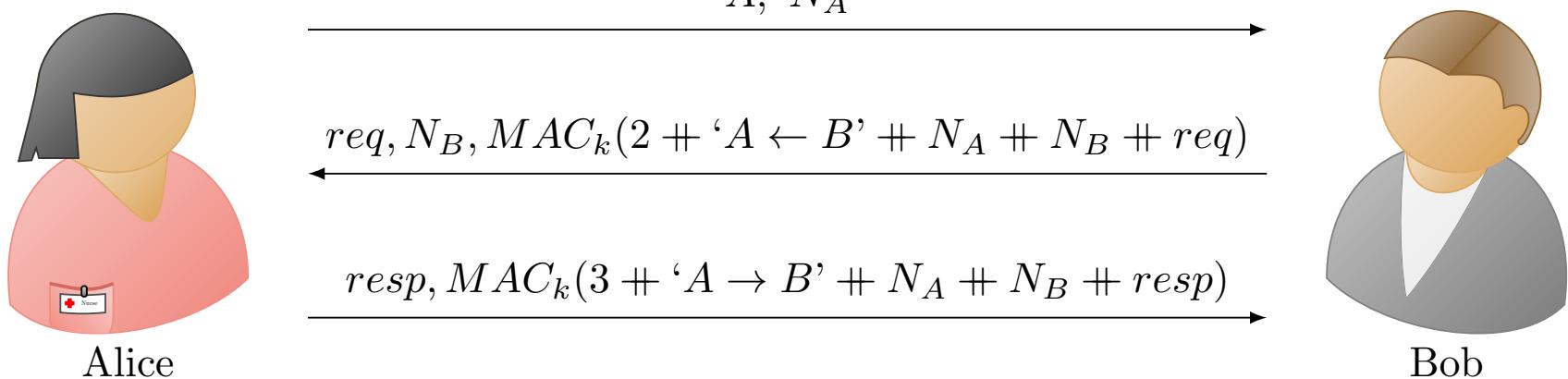
- Beside authenticating entities, these protocols authenticate the exchange of a request and a response between the entities.
- Required properties:
 - **Request authentication.**
 - The request was indeed sent by the peer.
 - **Response authentication**
 - The response was indeed sent by the peer.
 - **No replay.**
 - Every request/response was received at most the number of times it was sent by the peer.

Authenticated Request-Response Protocols

- Four variants:
 - 2PP-RR
 - 2RT-2PP
 - Counter-based-RR
 - Time-based-RR.
 - Key-exchange.

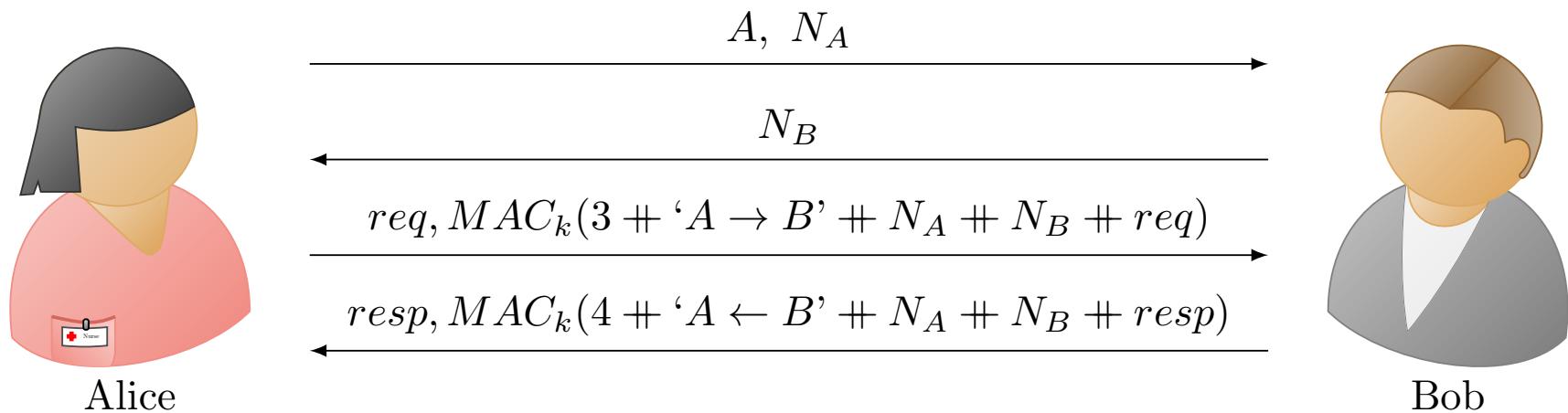
2PP-RR

- A three-flow nonce-based protocol.
- Significant drawback:
 - The request is sent by the responder and the initiator sends the response.
 - So initiator has to wait for a request rather sending it!!



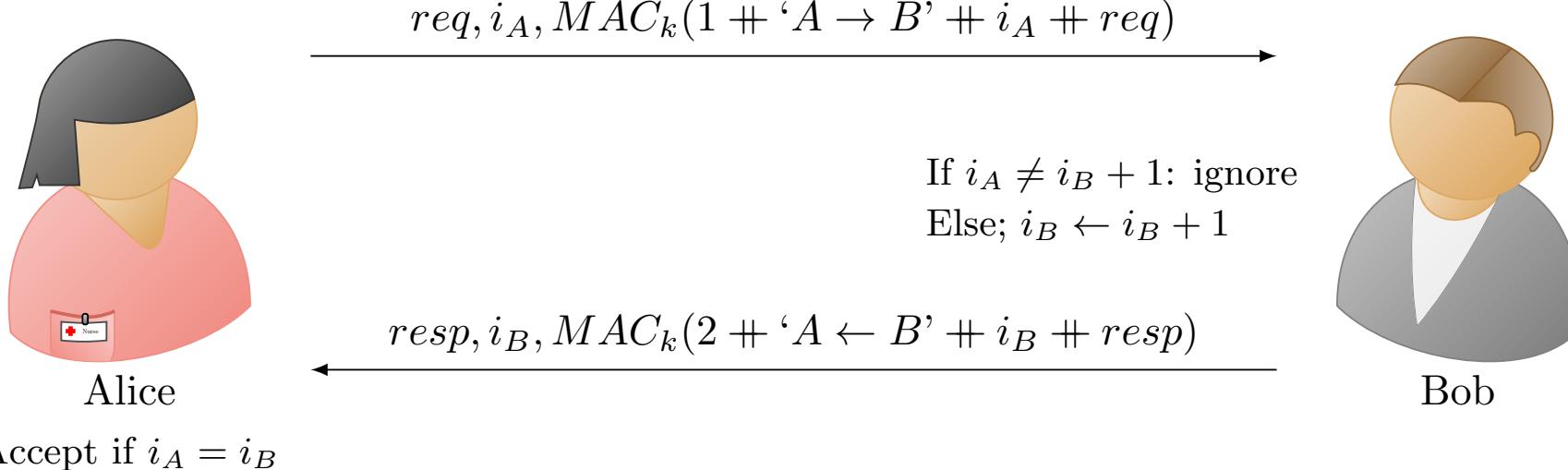
2RT-2PP

- A four-flow nonce-based protocol.
- Mainly fixes the drawback of 2PP-RT.



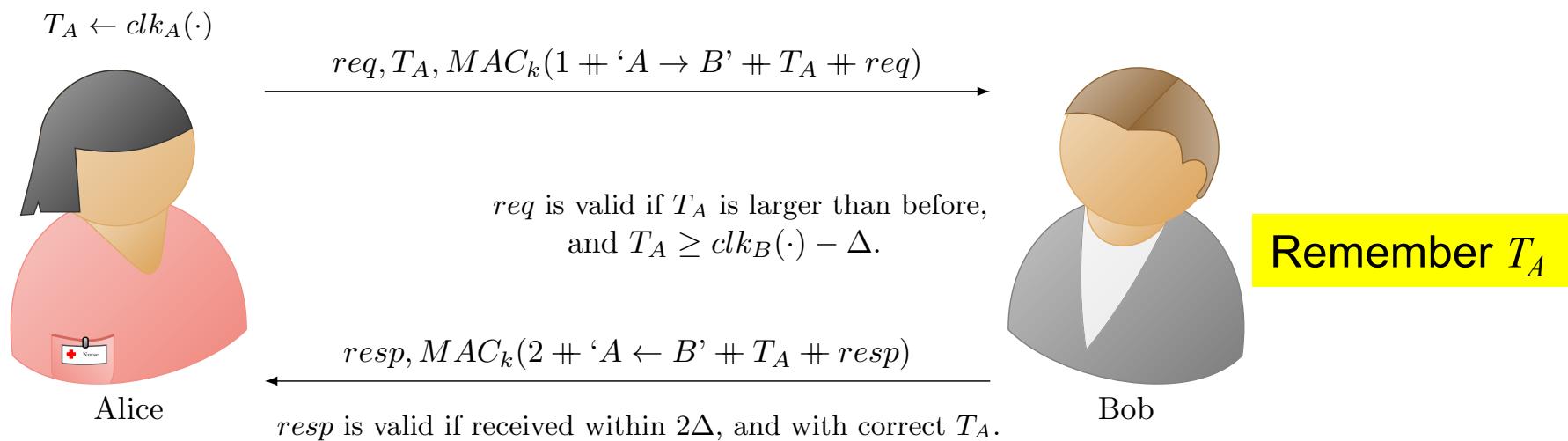
Counter-Based Authenticated RR

- **Simple stateful (counter) solution, requiring only one round:**
 - Unidirectional (run once for each direction if both are needed).
 - Parties maintain synchronized counter i of requests (and responses) to avoid replay attacks.
 - Recipient (e.g. Bob) validates counter received is $i + 1$
 - Both parties must remember counter



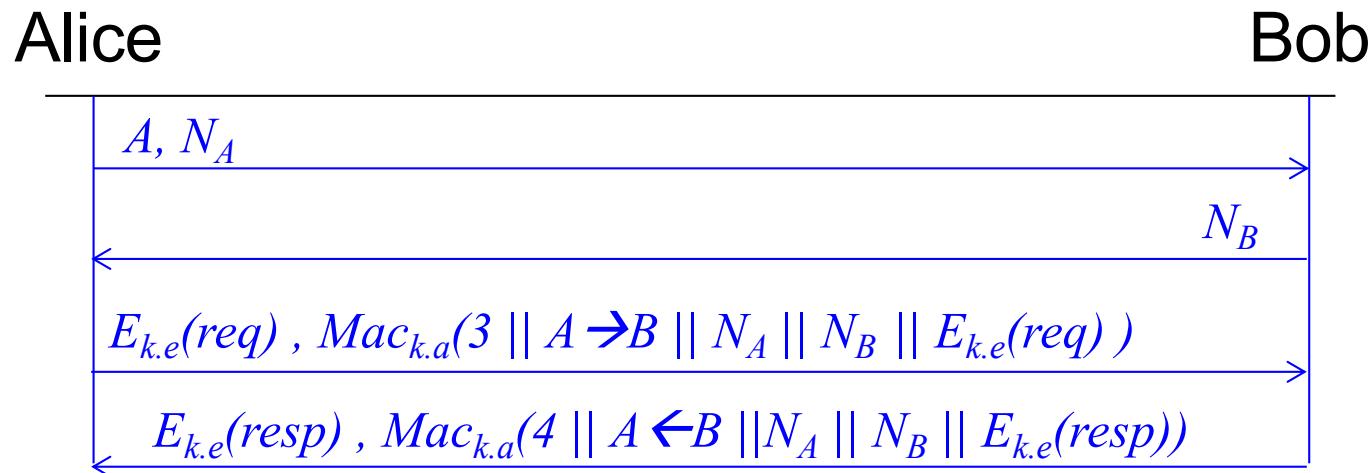
Time-Based Authenticated RR

- **Simple stateful (time) solution, requiring only one round:**
 - Use local clocks T_A, T_B instead of counters with two assumptions: bounded delays and bounded clock skews.
 - Responder (Bob):
 - Rejects request if: $T_B > T_A + \Delta$ where $\Delta \equiv \Delta_{skew} + \Delta_{delay}$
 - Or if he received larger T_A already
 - Maintains last T_A received, until $T_A + \Delta$
 - Initiator (Alice) does not need **any** state, when can Bob discard his?



2RT-2PP with Confidentiality

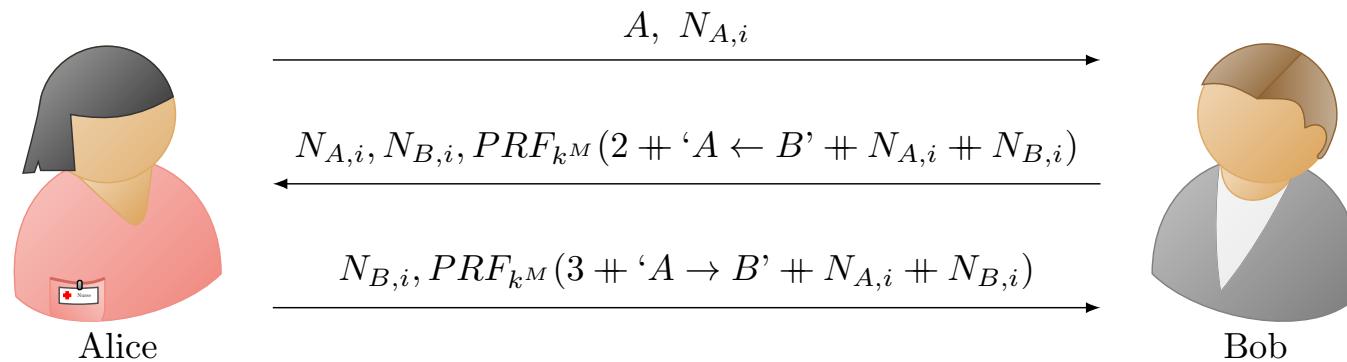
- **Secure connection: authentication, freshness, secrecy**
 - Independent keys: for encryption $k.e$, for authentication: $k.a$
 - How can we derive them both from a single key k ?
 - $k.e = PRP_k(\text{"Encrypt"})$, $k.a = PRP_k(\text{"MAC"})$
 - Hmm... same key encrypts all messages, in all sessions ☹
- **Can we improve security, by changing keys, e.g., btw sessions ?**



2PP Key Exchange Protocol

- Independent session keys, e.g. $k = \text{PRF}_{MK}(N_A, N_B)$
- Or, ‘directly’ for authentication and for encryption:
 $k.e = \text{PRF}_{MK}(\text{"Encrypt"}, N_A, N_B)$, $k.a = \text{PRF}_{MK}(\text{"MAC"}, N_A, N_B)$
- Improves security:
 - Exposure of session key does not expose (long-term) ‘master key’ MK
 - And does not expose keys of other sessions
 - Limited amount of ciphertext exposed with each session key k
- Later: reduce risk also from exposure of Master Key MK

Why a PRF is used instead of the MAC as before?



$$k_i^S = \text{PRF}_{k^M}(N_{A,i} + N_{B,i})$$

$$k_i^S = \text{PRF}_{k^M}(N_{A,i} + N_{B,i})$$

Key Distribution Centers (KDCs)

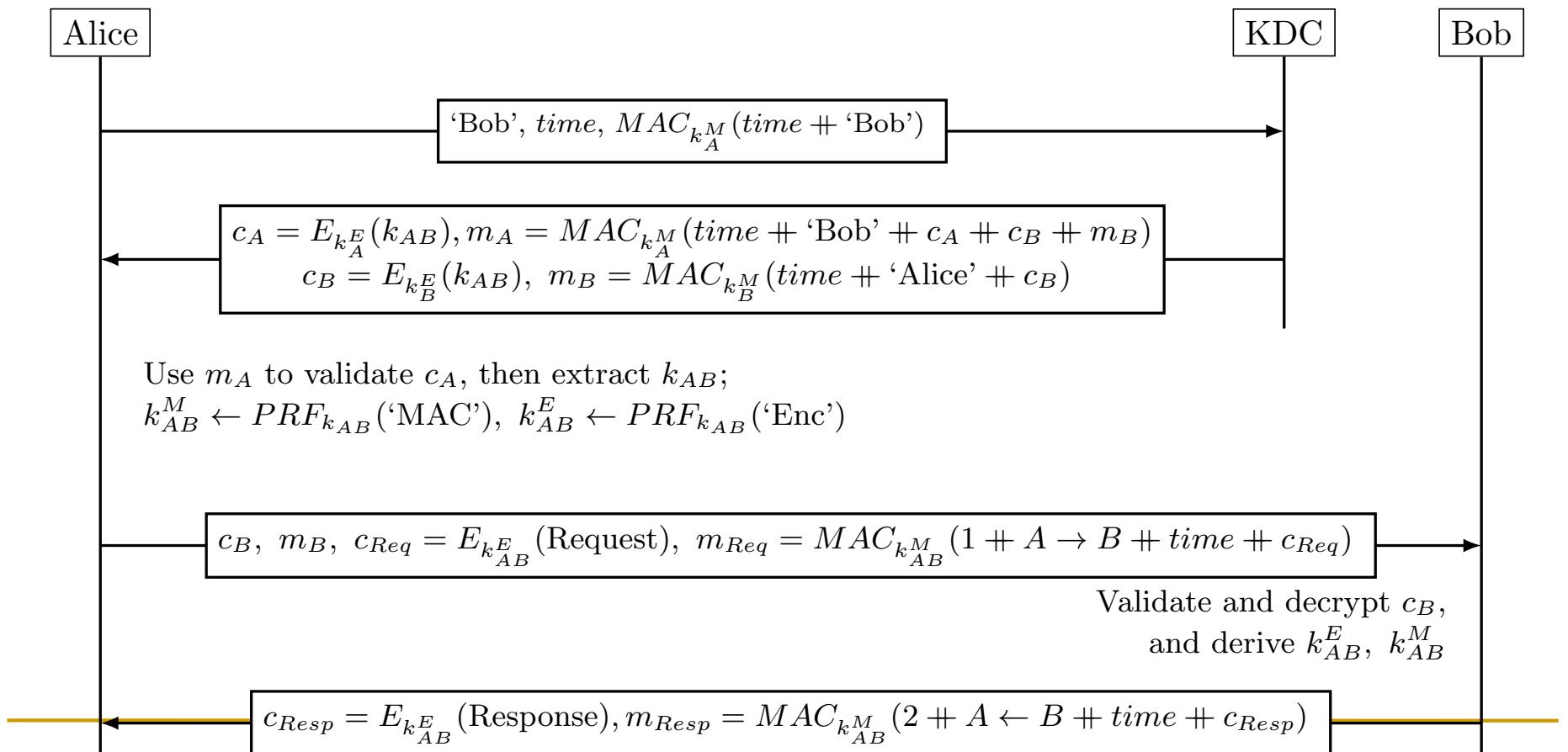
*Establish a shared key between two or more entities,
usually with the help of a trusted third party referred
to as KDC*

Key Distribution Center (KDC)

- Will focus on three party protocols; Alice, Bob, and KDC.
- KDC: shares keys with all parties ($k_A, k_B\dots$)
- Goal: help parties (A, B) establish k_{AB}
- We will study two protocols; simplified versions of:
 - The Kerberos protocol (secure) widely used in computer networks.
 - The GSM protocol (insecure) used by cellular networks.

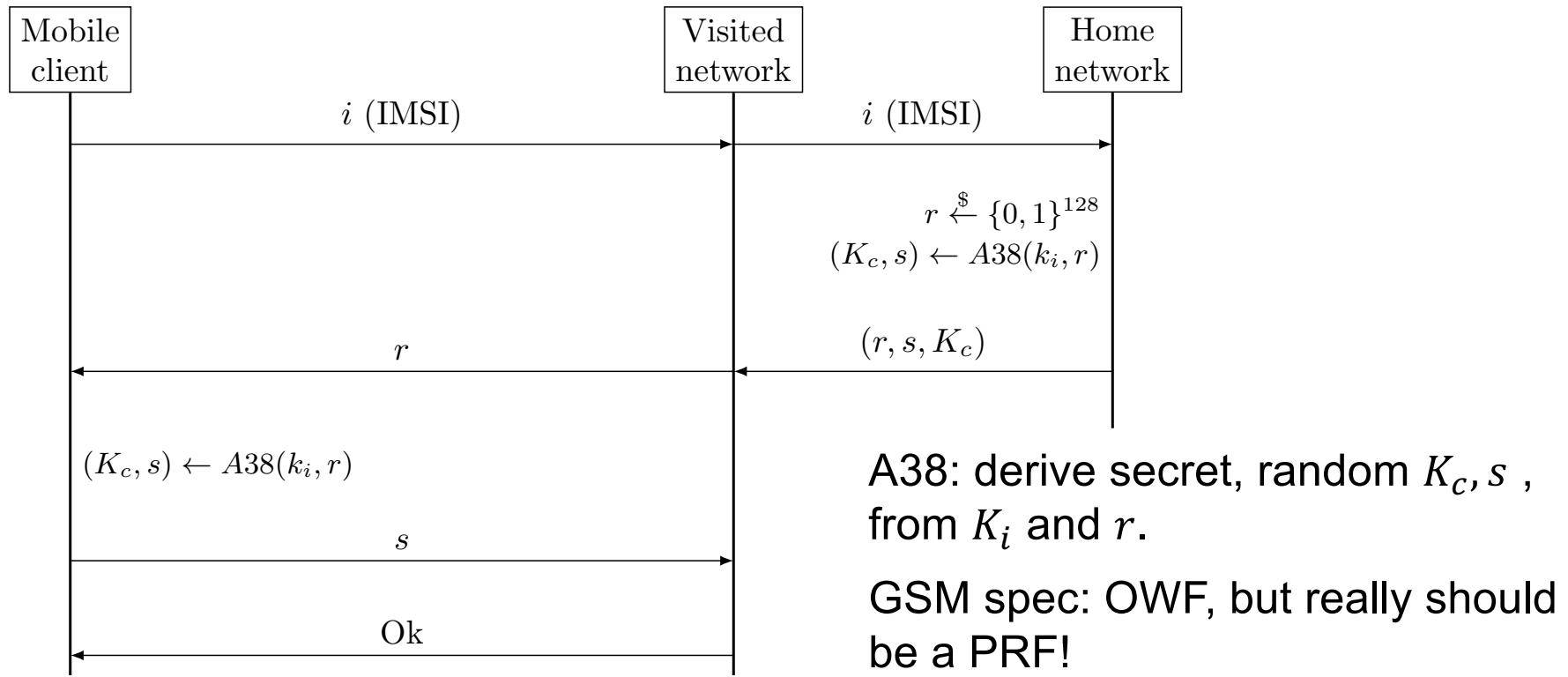
The Kerberos KDC Protocol

- KDC shares keys k_A^E (enc.), k_A^M (MAC) with Alice and k_B^E , k_B^M with Bob
- Goal: Alice and Bob share k_{AB} , then derive: k_{AB}^E , k_{AB}^M
- KDC performs access control as well; controlling whom Alice can contact.

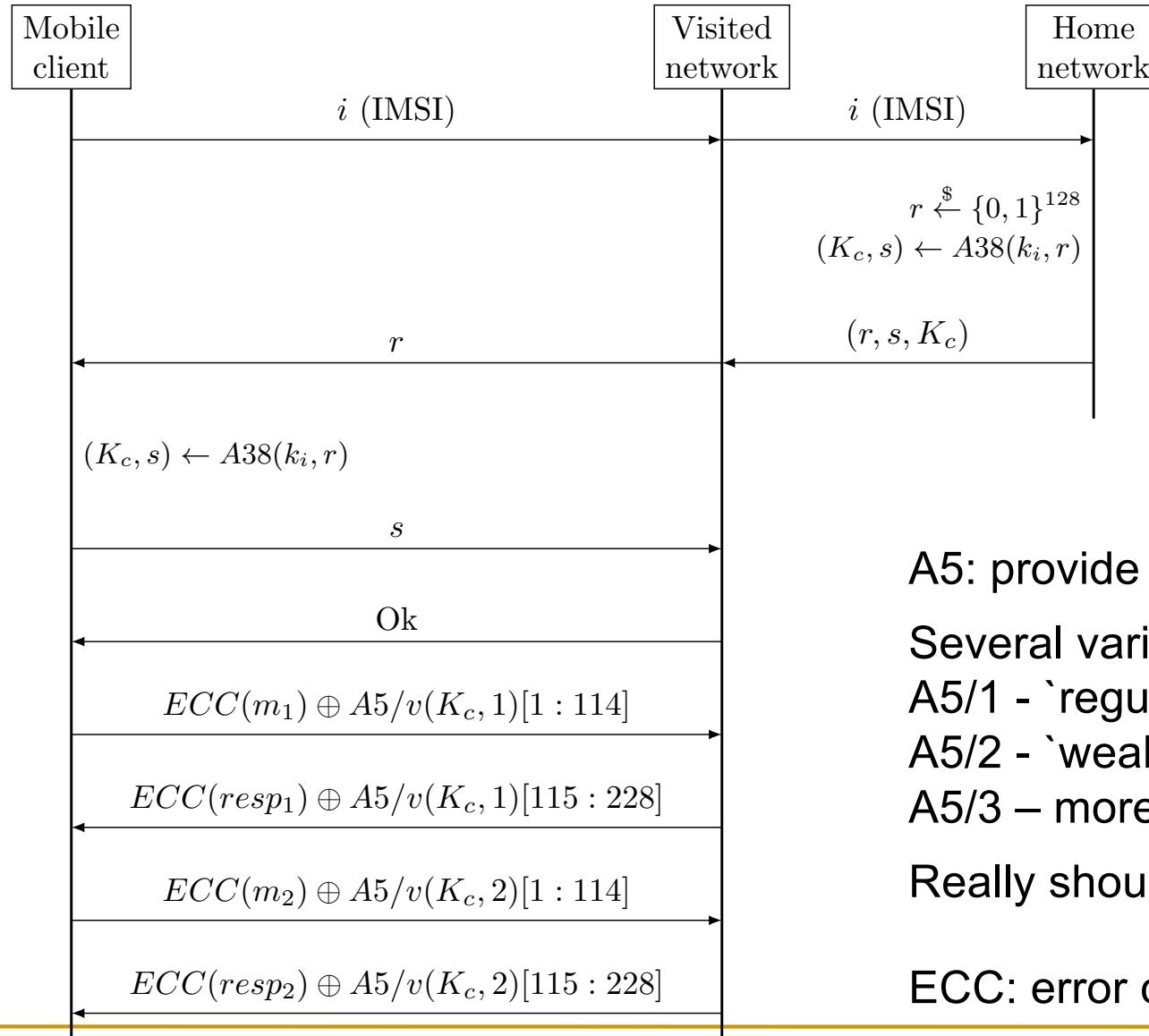


The GSM Handshake Protocol

- Mobile client
 - Identified by i (IMSI: International Mobile Subscriber Identifier)
- Visited network (aka Base station); not fully trusted !
- Home network; trusted, shares key K_i with client i



Example – Sending two messages



A5: provide ‘pad’ for encryption

Several variants:

A5/1 - ‘regular’

A5/2 - ‘weak’

A5/3 – more secure

Really should be a PRF!

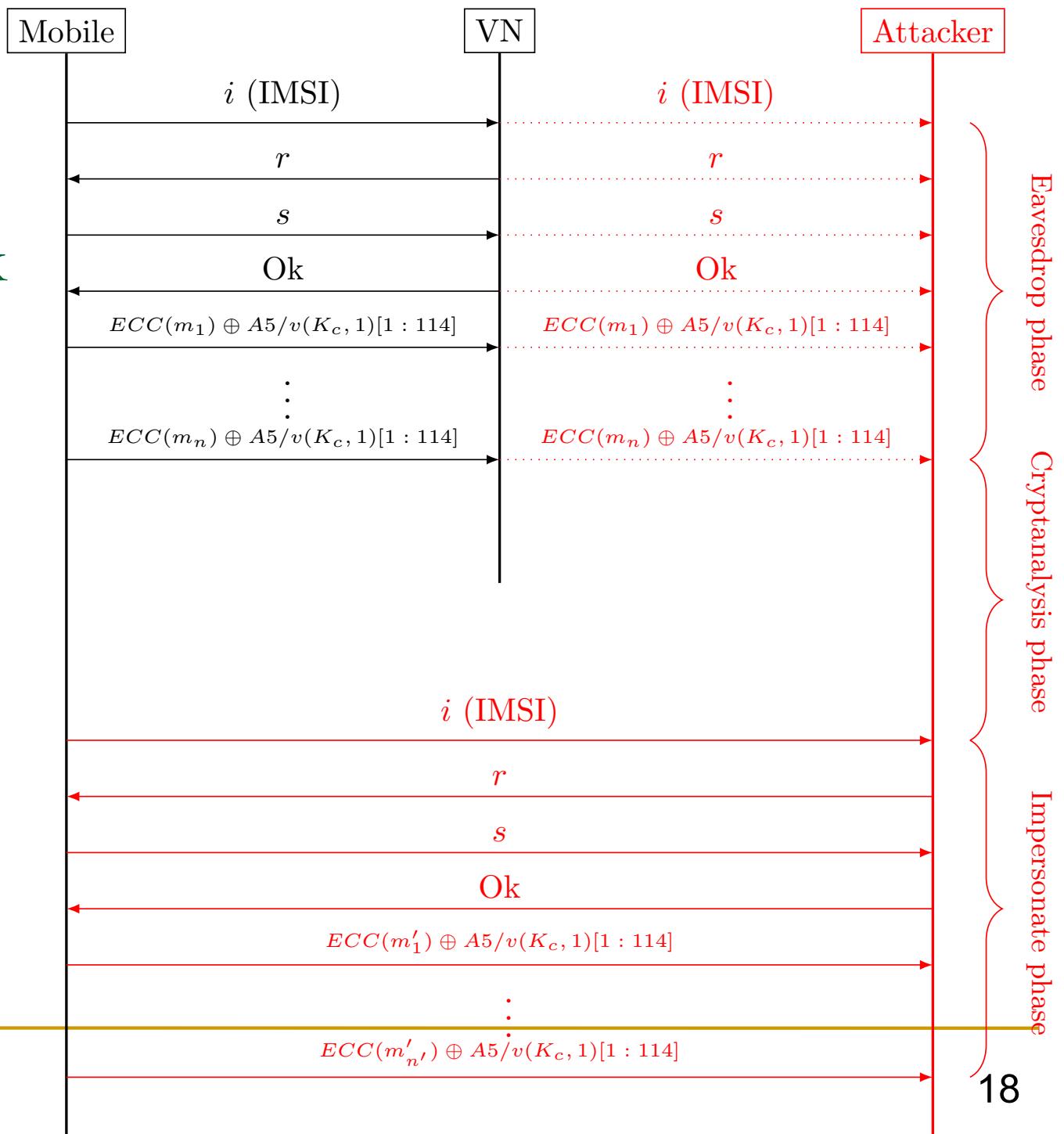
ECC: error correcting code.

Attacks on GSM

- We will explore two such attacks:
 - Visited network impersonation replay attack.
 - Downgrade attack.

Visited-network Impersonation Attack

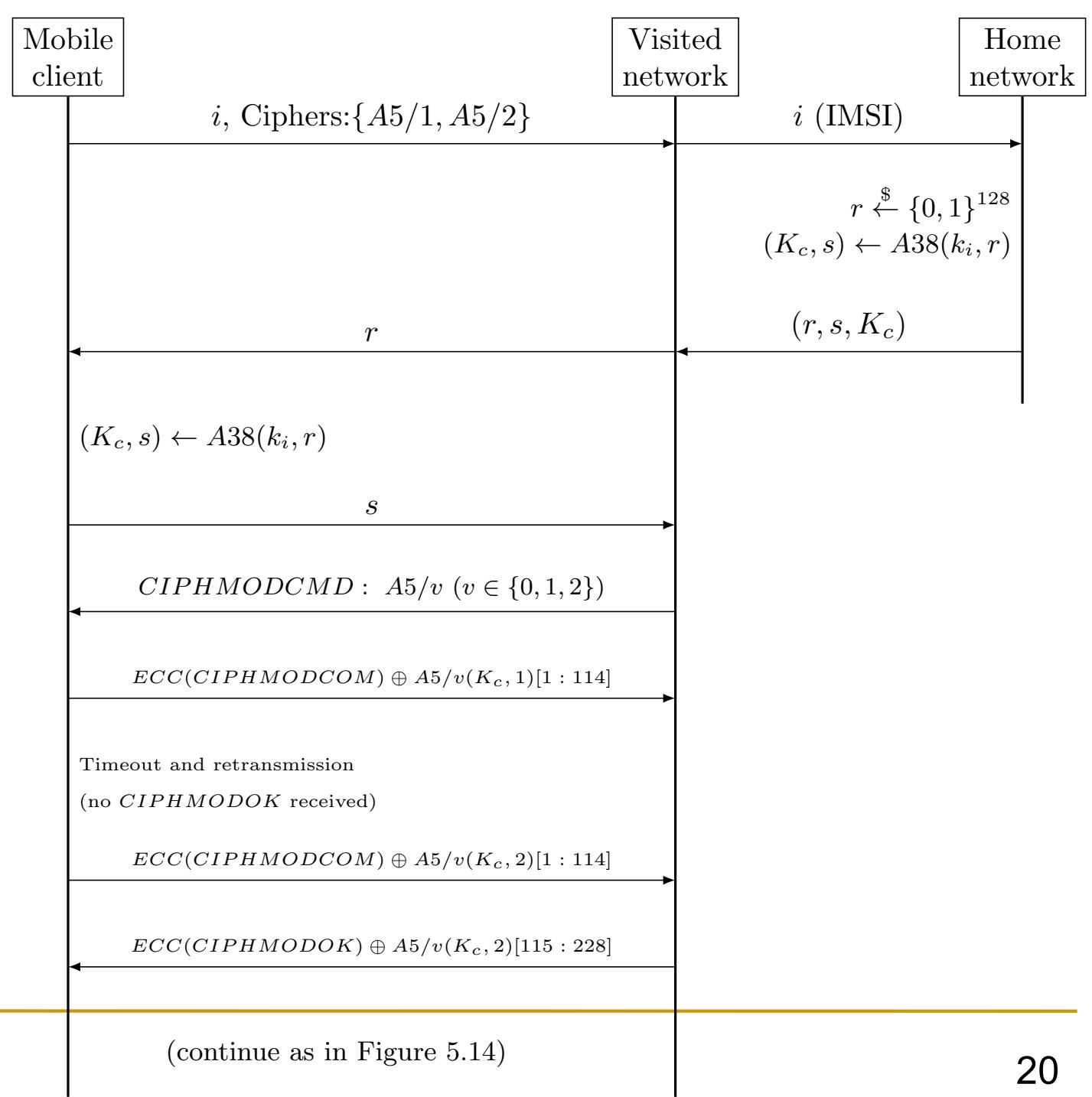
Note: does NOT
Impersonate **mobile**,
only Visited network.



GSM Ciphersuites Downgrade Attack

- A ciphersuit is the set of cryptographic schemes used in a protocol execution.
- Ciphersuit negotiation:
 - Mobile sends list of cipher-suites it supports
 - Visited-net selects best one that it also supports
- GSM encryption algorithms E_k :
 - A5/0: none, A5/1: broken, **A5/2: useless (break with only 1sec of ciphertext!)**, A5/3: ‘other’
- A MitM attacker may trick these parties to use a weak suite although the parties can support a stronger one.
- Let’s first see how ciphersuite negotiation happened in GSM.

GSM Handshake, With Cipher- negotiation.



Cipher mode messages, negotiation

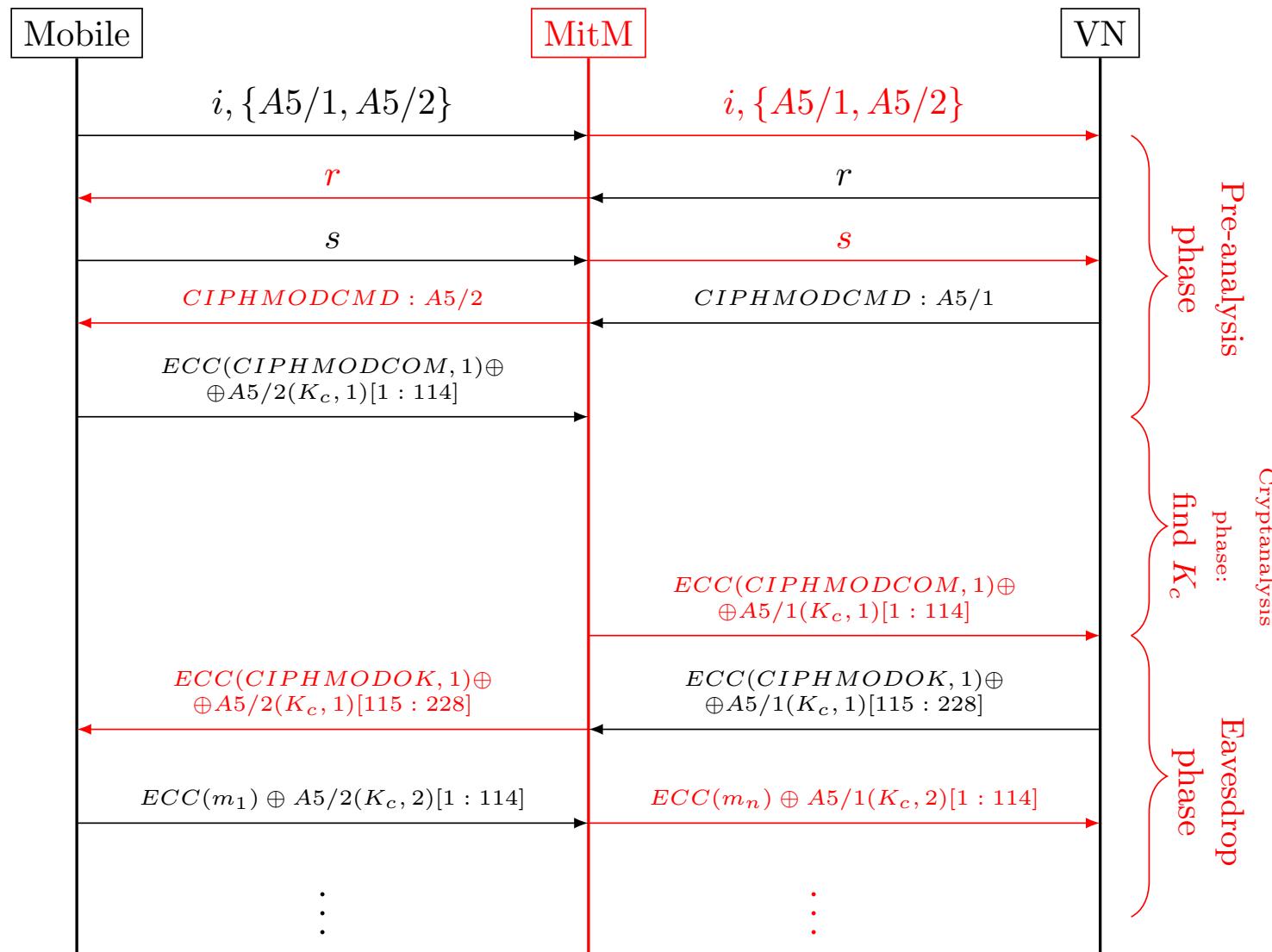
- Mobile sends list of supported ciphers
- VN sends choice in: **CIPHMODCMD**
 - **Cipher Mode Command**
- Mobile confirms by sending encrypted:
CIPHMODCOM: cipher mode complete
 - If not received (in few msecs), VN disconnects
- VN Acks: **CIPHMODOK: cipher mode Ok**
 - If not received, mobile resends **CIPHMODCOM**

GSM ciphersuite facts: for fun and profit

- GSM derives, uses same K_c for all ciphers
- CTO attack on A5/2 requires 900 bits, 1 sec
 - If ciphertext is after GSM's ECC, of course
 - Lots of redundancy
- Visited networks don't downgrade to A5/2
- Mobile encrypts, sends CIPH**MOD**COM
 - Resends (in few msecs) if no CIPH**MOD**OK
 - New encryption each time (counter)
 - 456bit message (after ECC)
- Allow 12s delay for the s message

Simplified Downgrade Attack

Efficient attack known only for A5/2; Client, Visited-net normally prefers A5/3 or A5/1, which are harder to break. Attack forces use of A5/2 !!

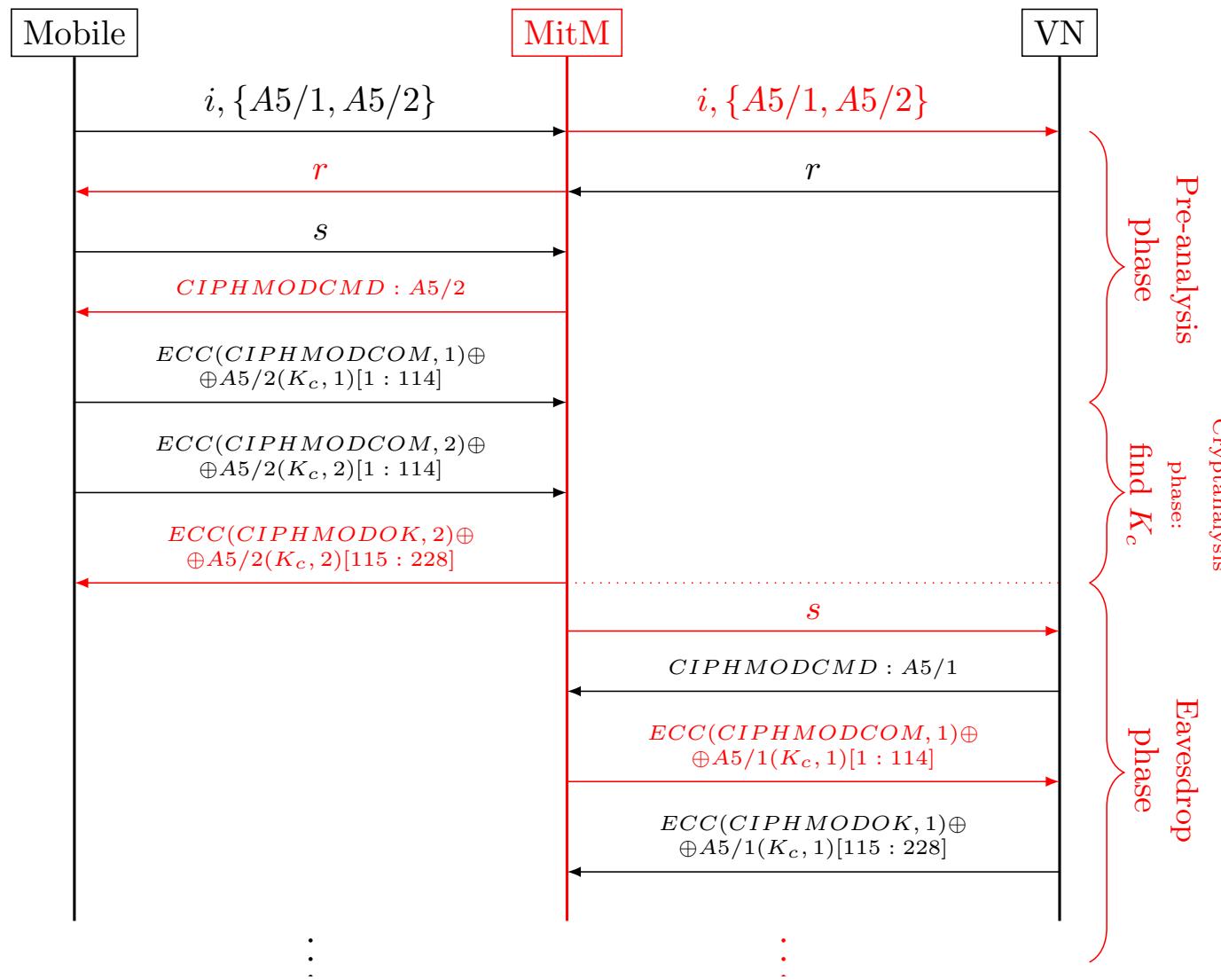


Simplified downgrade attack - Fails

- Fails in practice due to two reasons:
 - VN would time-out since CIPH**MODCOM** is not received in few milliseconds
 - A5/2 CTO attacker requires a second to reveal the key.
 - And CIPH**MODCOM** is only 456 bits
 - A5/2 CTO attacker requires 900 bits.

Real Downgrade Attack

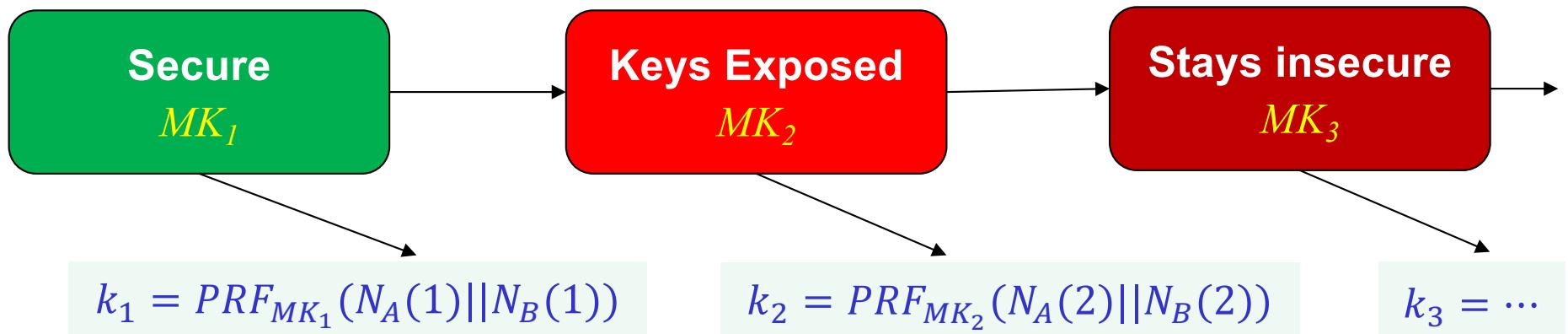
Works even if VN insists to use A5/1; attacker tricks client to use A5/2.
That suffices, since GSM uses same key for all cryptosystems!



Improving Resiliency to Key Exposure

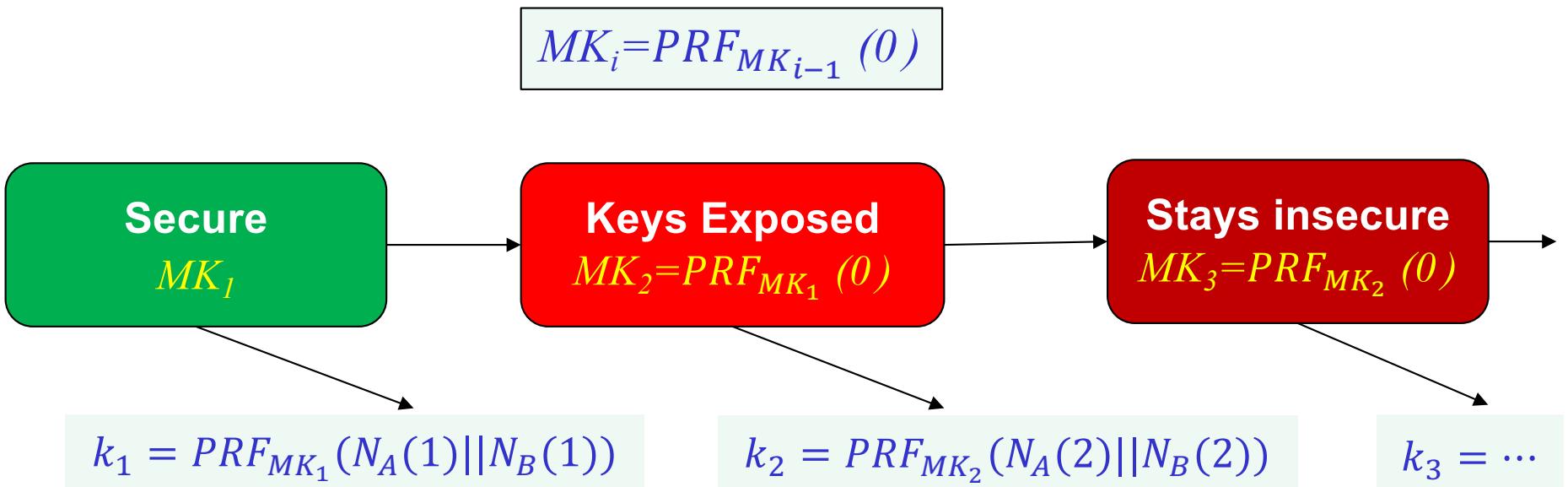
Forward Secrecy I

- **So far:** session key $k_i \not\Rightarrow k_j$ (expose no other keys)
 - And master key was fixed for all sessions
- **Idea:** we can do better!
 - Change the master key each session: MK_1, MK_2, \dots
- **Forward Secrecy (FS):** master key $MK_i \not\Rightarrow k_j (j < i)$
 - I.e., MK_i (and k_i) don't expose keys, communication of previous sessions ($j < i$)



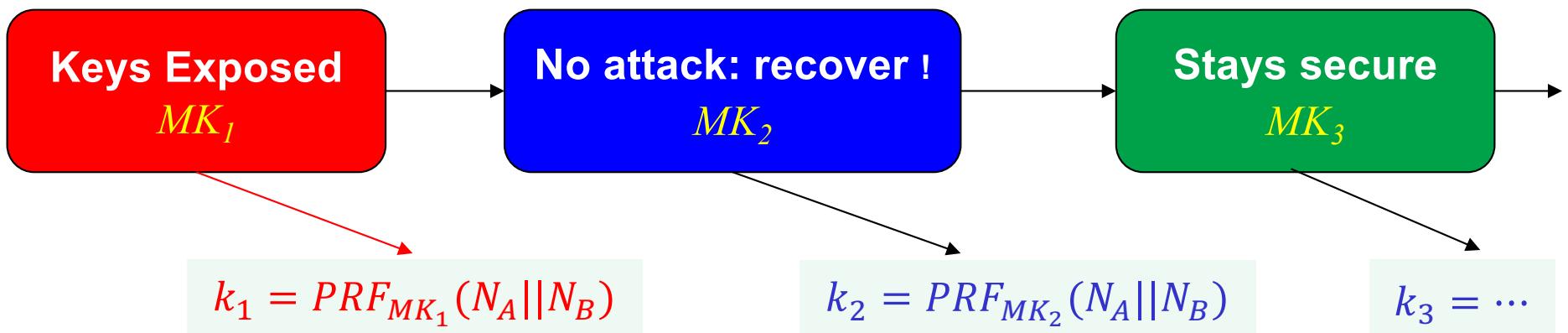
Forward Secrecy II

- **Forward Secrecy (FS):** master key $MK_j \not\Rightarrow k_i (j > i)$
 - Session i is secret even if any state of later sessions is exposed.
 - Uni-directional: $MK_i \rightarrow MK_{i+1}$, but $MK_{i+1} \not\rightarrow MK_i$
 - **How?** Solution: PRF!



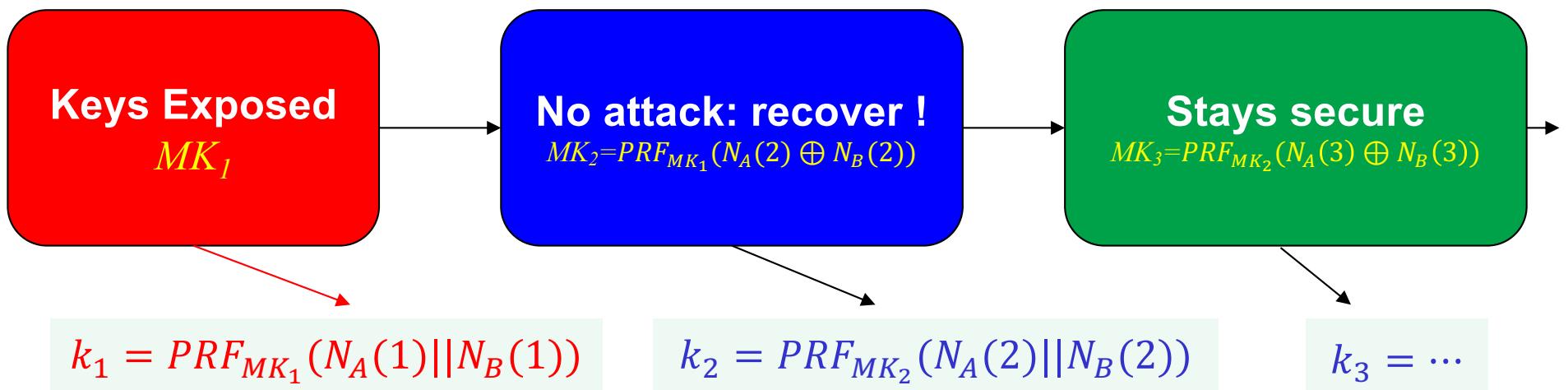
Recover Security

- Can we also **recover** security?
 - MK_{i_R-1} exposed, yet $MK_{i_R}, MK_{i_R+1} \dots$ secure ?
 - Idea: assume **no attack** during ‘recovery session’ i_R



Recover Security (RS)

- Recover security: session i secure if :
 - session i is secure if it's keys are not given to attacker, and either session $i - 1$ is secure, or there is no attack during session i
 - How? The RS-Ratchet Protocol:
 - Let $N_A(i), N_B(i)$ denote session's i nonces
 - Then: $MK_i = \text{PRF}_{MK_{i-1}}(N_A(i) \oplus N_B(i))$

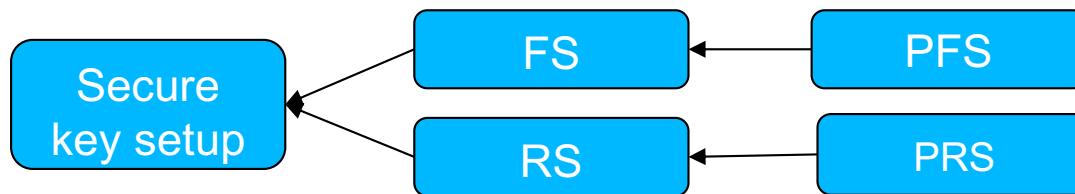


Stronger Notion of Resiliency

- **Perfect Forward Secrecy (PFS):** session i is secure even if attacker is given, only after session i ends, all keys of all other sessions, and Master Key of session i
 - *All include future and past sessions.*
- **Perfect Recover Security (PRS):** session i is secure if it's keys are not given to attacker, and either session $i - 1$ is secure, or there is no MitM attack during session i
- How? **public-key** (key exchange) protocols – next topic!

Resiliency Notions: Shared + Public Key

Notion	Session i is secure if keys not exposed, and...	Crypto
Secure key-setup	... attacker is given <u>session key</u> k_j ($j \neq i$). Master key never exposed !	Shared key
Forward secrecy	... attacker is given all keys of sessions $> i$	Shared key
Perfect Forward Secrecy (PFS)	... is also given all keys of sessions $< i$, <u>but only after session i ended</u>	Public key
Recover Security	... if no attack during session i , or if previous session, $i - 1$, is secure	Shared key
Perfect Recover Security (PRS)	... if no MitM attack during session i , or if previous session, $i - 1$, is secure	Public key



Covered Material From the Textbook

- Chapter 5
- Sections 5.3, 5.4, 5.5, and 5.6

Thank You!

