The Kerberos Authentication System
Course Outline

Technical Underpinnings

- authentication based on key sharing
- Needham-Schroeder protocol
- Denning and Sacco protocol

Kerbeors V 4

- Login and client-server authentication
- Credential establishment and cache
- Key Version Numbers
- The KDC Database
- Interrealm Authentication
- Data Encryption
- Data Integrity
- Kerberos V 4 Message Formats

Kerbeors V 5

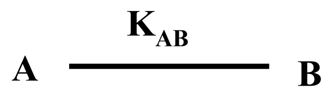
- ASN.1 Data Representation Language
- Delegation of Rights
- Ticket Lifetimes
- Key Version Numbers
- Interrealm Hierarchy
- Preauthentication
- KDC Database
- Double TGT Authentication
- Data Encryption / Integrity
- Kerberos V 5 Message Formats and Protocol Flows

Kerberos Future Developments and Use

Kerberos V4

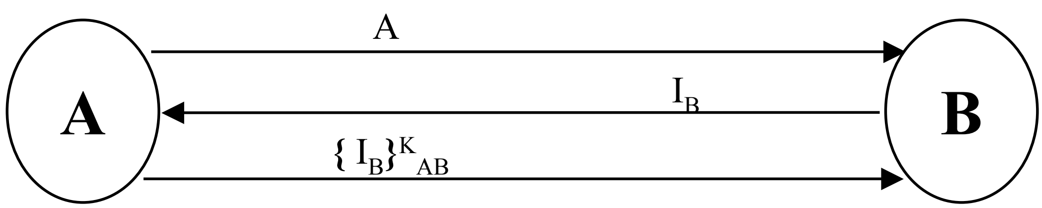
Technical Underpinnings and Description

Authentication Based on Secret-Key Sharing

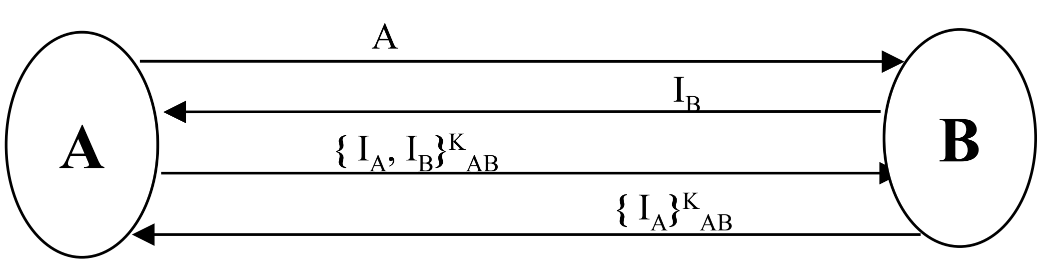


A and B share secret key $\mathbf{K_{AB}}$

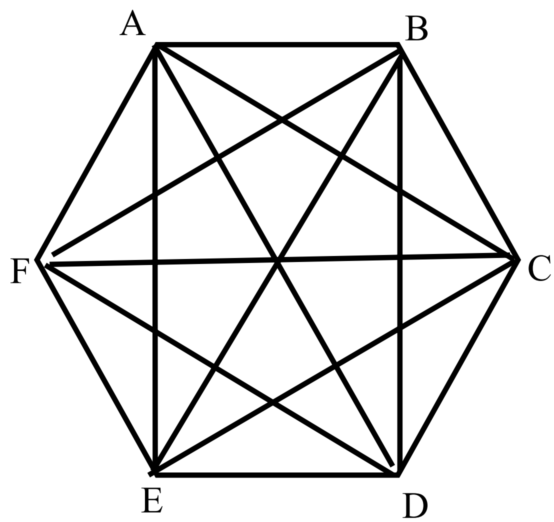
One-way authentication (?)



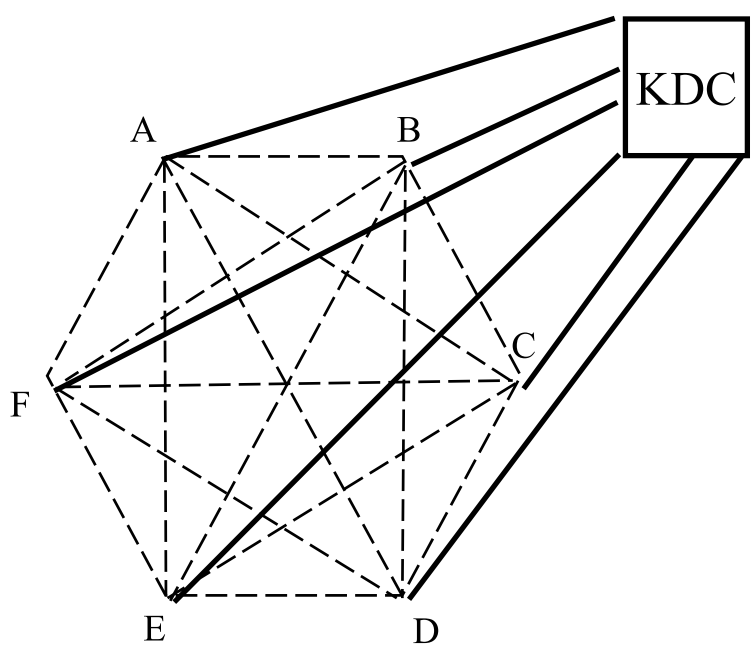
Two-way (mutual) authentication



Pairwise Authentication - O(n^2) keys



Trusted Third-Party Authentication - O(n) keys



—— shared *long-term* key (e.g., 6 mos.)

········· shared *session* key (e.g., 8 hours)

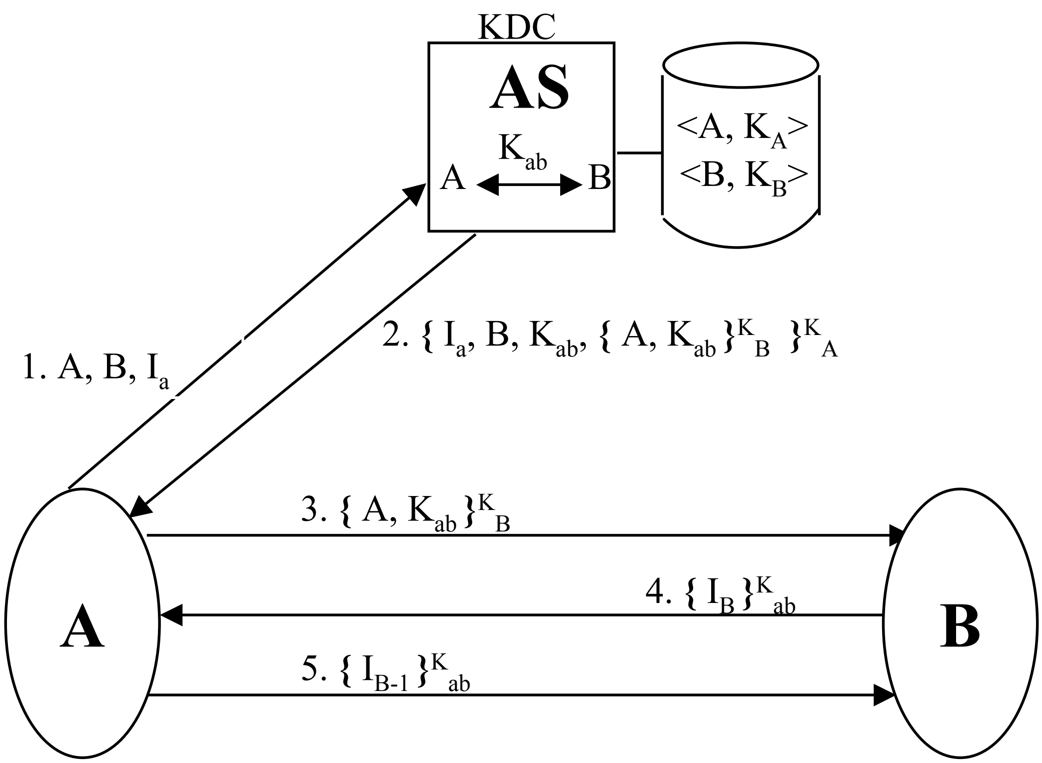
KDC Key Distribution Center

Needham - Schroeder's Protocol (1978)

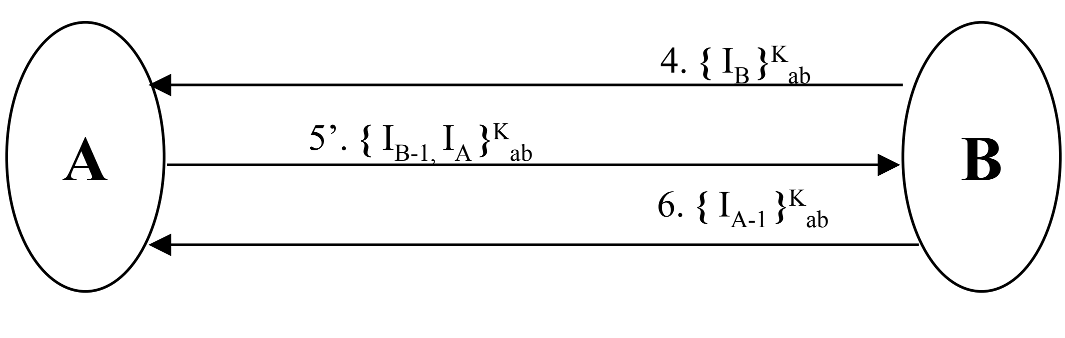
A = initiator peer, client;
K_A = A's private, long-term, key
I_a , I_A = A's nonces (challenges)

B = recipient peer, server;
K_B = B's private, long-term, key
I_B = B's nonce (challenge)

KDC (**AS**) = Authentication Server



Steps 1 - 3 : distribution of session key K_{ab}
Steps 4, 5 : *one-way authentication*; i.e., B authenticates A



Steps 4 - 6 : *two-way(mutual) authentication* of A and B

Needham - Schroeder's Protocol (ctnd.)

1. What if I_a is not used in messages 1, 2 ?

Intruder X can replay an old AS response to A's request

1. A, B
2. $\{ B, K_{old\text{-}ab}, \{ A, K_{old\text{-}ab} \}^{K_B} \}^{K_A}$

- forces the reuse of an **old session key** past the key's lifetime

2. What if identity B is not used (encrypted) in message 2 ?

Registered user X can masquerade as B, and can make A believe it is communicating with B

- changes B to X in message 1.
- intercepts messages 3, 5 and generates correct responses 4, 6.

1. A, X
2. $\{ B, K_{ax}, \{ A, K_{ax} \}^{K_X} \}^{K_A}$
3. $\{ A, K_{ax} \}^{K_x}$
4. ....

3. What if A repeatedly requests a session with B from AS ?

A obtains known plaintext-ciphertext pairs $< K^i_{ab}, \{ A, K^i_{ab} \}^{K_B} >$, i= 1,..., n and performs cryptanalysis to discover B's secret key K_B.

Countermeasures: (1) replace $\{ A, K^i_{ab} \}^{K_B}$ with $\{ TK_i \}^{K_B} \{ A, K^i_{ab} \}^{TK_i}$ where TK_i is a temporary key unknown to A.

(2) use $\{ \textit{confounder}_i, A, K^i_{ab} \}^{K_B}$ instead of $\{ A, K^i_{ab} \}^{K_B}$ where *confounder*$_i$ is a (pseudo) random number.

4. What if intruder X discovers K_{ab} (but not K_A or K_B)?

Intruder X can masquerade as A, and can make B believe it is communicating with A

- replays message $\{ A, K_{ab} \}^{K_B}$
- knows f = I_B - 1, and generates correct response 5.

This vulnerability was pointed out by Denning and Sacco in 1981

Denning and Sacco's Protocol (1981)

Same assumptions as Needham's and Schroeder's.

In addition, T = timestamp is generated by AS, and all clocks are *tightly* synchronized; i.e.,

$$|CLOCK_i - T| < \Delta t_1 + \Delta t_2,$$

for all i = A, B, and where Δt_1 = discrepancy between local clocks and AS' clock
Δt_2 = network delay

1. A -> AS : A, B

2. AS -> A : $\{ B, K_{ab}, T, \{ A, K_{ab}, T \}^{K_B} \}^{K_A}$

3. A -> B : $\{ A, K_{ab}, T \}^{K_B}$

4. B -> A : $\{ I_B \}^{K_{ab}}$

5. A - > B : $\{ I_{B-1} \}^{K_{ab}}$

Limited lifetime of $\{ A, K_{ab}, T \}^{K_B}$ has the following consequences:

- the ticket $\{ A, K_{ab}, T \}^{K_B}$ cannot be replayed (or reused)
- an intruder that discovers $\mathbf{K_{ab}}$ cannot masquerade as **A**

However,

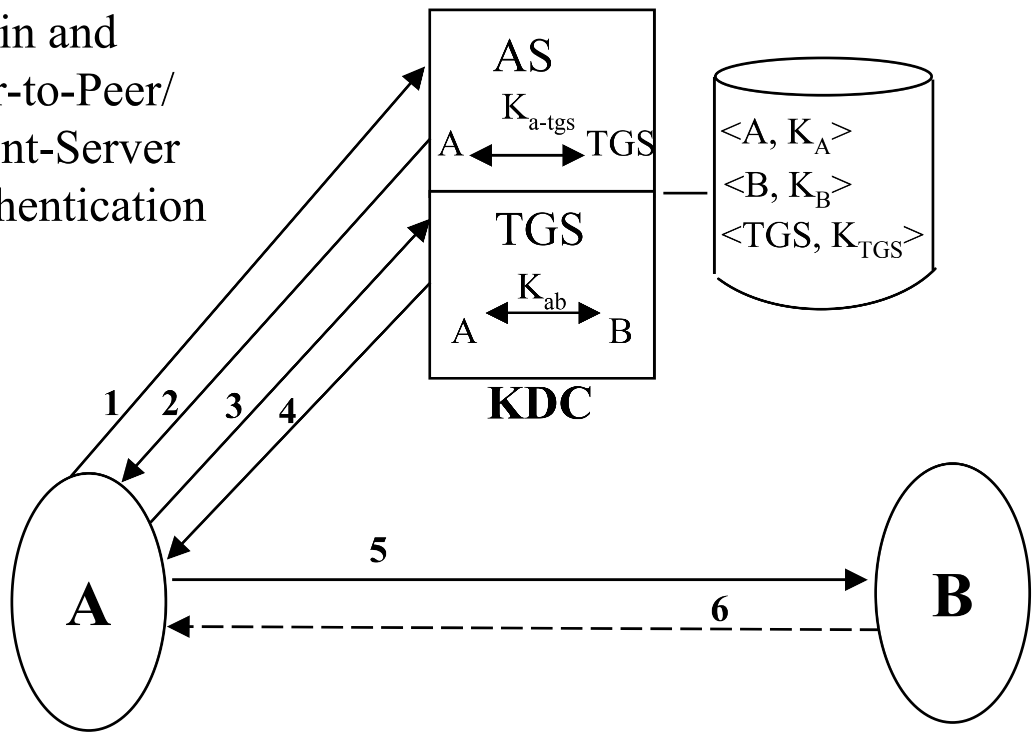
- network delays or out-of-synch local clocks can cause denial of service

and

- lifetime limit for K_{ab} cannot be enforced by ticket $\{ A, K_{ab}, T \}^{K_B}$ (no lifetime limit)
- ticket $\{ A, K_{ab}, T \}^{K_B}$ cannot be cached and reused by A.

Kerberos V4 (MIT 1987 - 1992)

Login and
Peer-to-Peer/
Client-Server
Authentication



1. **AS_REQ** : A, T_{a1}, $lifetime_1$, TGS

2. **AS_REP**: A, T_{a1}, $expr\_time_1$, $\{ K_{a\text{-}tgs}, TGS, expr\_time_1, \{ Ticket_{a\text{-}tgs} \}^{K_{TGS}}, T_{a1} \}^{K_A}$
   where $Ticket_{a\text{-}tgs} = < A, @A, K_{a\text{-}tgs}, lifetime_1, T_{kdc1}, TGS >$

3. **TGS_REQ** : $\{ Ticket_{a\text{-}tgs} \}^{K_{TGS}}$, $\{ authenticator_{a\text{-}tgs} \}^{K_{a\text{-}tgs}}$, T_{a2}, $lifetime_2$, B
   where $authenticator_{a\text{-}tgs} = < A, checksum_1, T_{a2} >$

4. **TGS_REP** : A, T_{a2}, $expr\_time_2$, $\{ K_{ab}, B, expr\_time_2, \{ Ticket_{ab} \}^{K_B}, T_{a2} \}^{K_{a\text{-}tgs}}$
   where $Ticket_{ab} = < A, @A, K_{ab}, lifetime_2, T_{kdc2}, B >$

5. **AP_REQ** : $\{ Ticket_{ab} \}^{K_B}$, $\{ authenticator_{ab} \}^{K_{ab}}$ (for *one-way* authentication)
   where $authenticator_{ab} = < A, checksum_2, T_{a3} >$

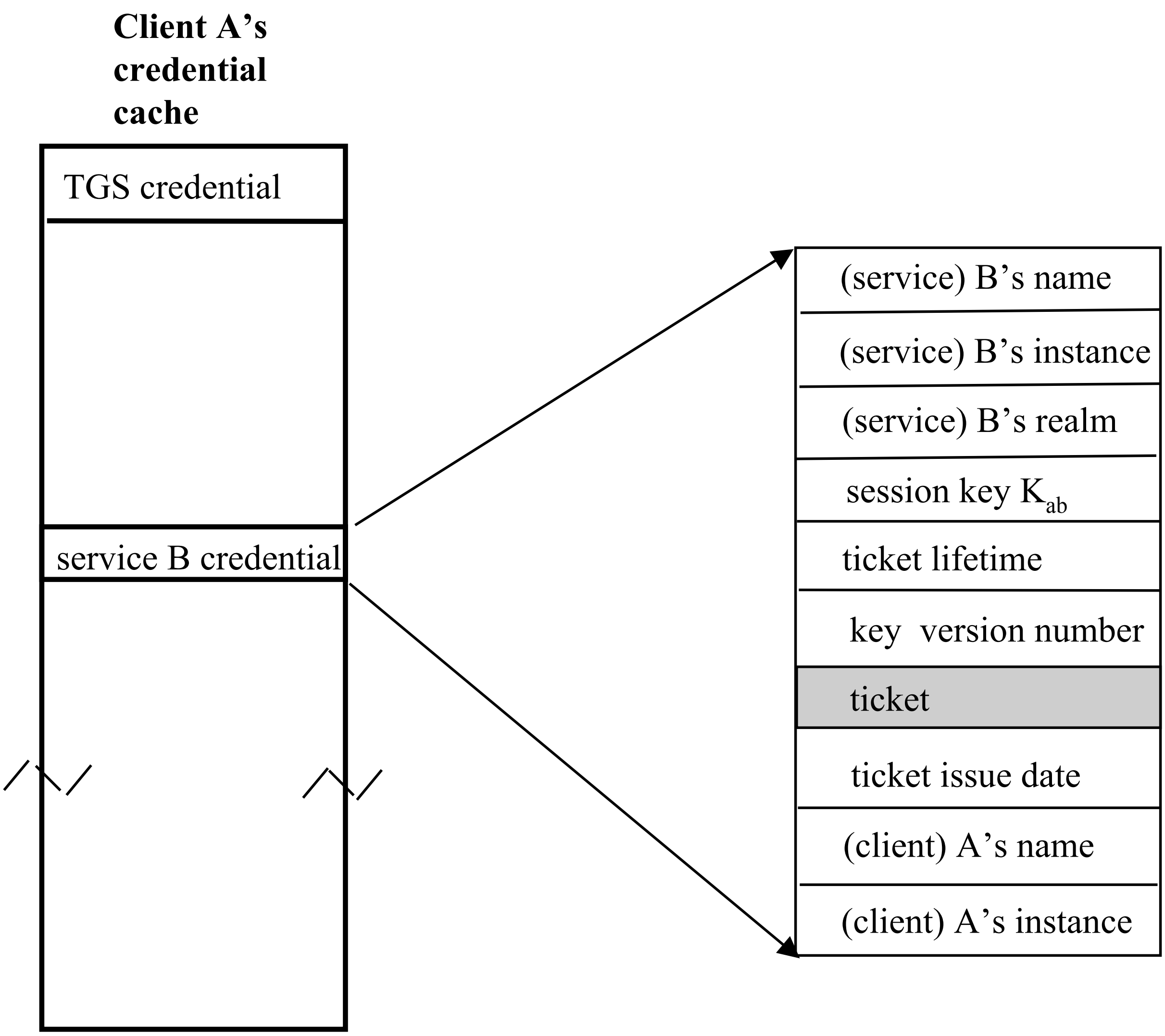
6. **AP_REP** : $\{ checksum_2 + 1 \}^{K_{ab}}$ = OPTIONAL (for *mutual* authentication)

Credential Establishment and Cache

Credential cache is held in a file *accessible only by the user's processes*.

Cache entries are filled by the execution of *messages 1 - 4* of Kerberos.

Cache entry structure returned by "get_cred".



Key Version Numbers (krb v 4)

Motivation: Both users and servers change their keys over time.
(e.g., passwords, server keys).
Outstanding tickets may exist which are encrypted with old key.

Unless servers remember old keys, communication fails.

Failed communication cannot always be reinitiated
(e.g., batch applications fail).

Approach: Maintain a *version number* for each key **.**

Servers' responsibility to save keys with older version numbers.

Tickets and protocol messages only include the expected
key version number.

Maximum number of old keys do not typically exceed two to three.
(max. life of a K V4 ticket is about 21 hours plus max. KDC update delay; exception: *long-life patches* allowing one-month tickets)

Limitation: Password updates may not propagate to all slaves instantaneously.

User logins transparently directed to a KDC slave may fail for a
until password updates propagate to KDC slaves.

Users must remember previous password (e.g., previous version).

Network-Layer Addresses in Tickets

Motivation: Theft of credential cache entries (i.e., tickets and corresponding session keys)
use of stolen tickets and session keys from foreign network locations

Situation: unattended workstations, root privileges to someone else's system

Note: Theft of tickets and authenticators *alone* by an intruder does not give the intruder a ticket's session key
Nevertheless, theft of tickets and authenticators can be a threat for all applications that do not use the session key and detect attack beyond initial authentication.

Approach: Place ticket user's network-layer (e.g., IP) address in ticket.
(Why not in authenticator ?)

Limitations: Approach disallows legitimate delegation of credentials.
Network-layer addresses can be faked without great difficulty.

KDC Replication

Motivation: Avoid *single point of failure* and *performance bottleneck*

Approach: Maintain a single Master KDC and multiple Slave KDCs.
Master KDC is Readable / Writeable whereas Slave KDCs are Read-only.
Slave KDCs are updated periodically by Master KDC, or by administrative command.
Unencrypted file containing Master KDC database is downloaded to each Slave KDC

Reason: Most KDC operations require Read-only access
KDC updates are typically required for infrequent operations; e.g., add / delete users, change passwords.

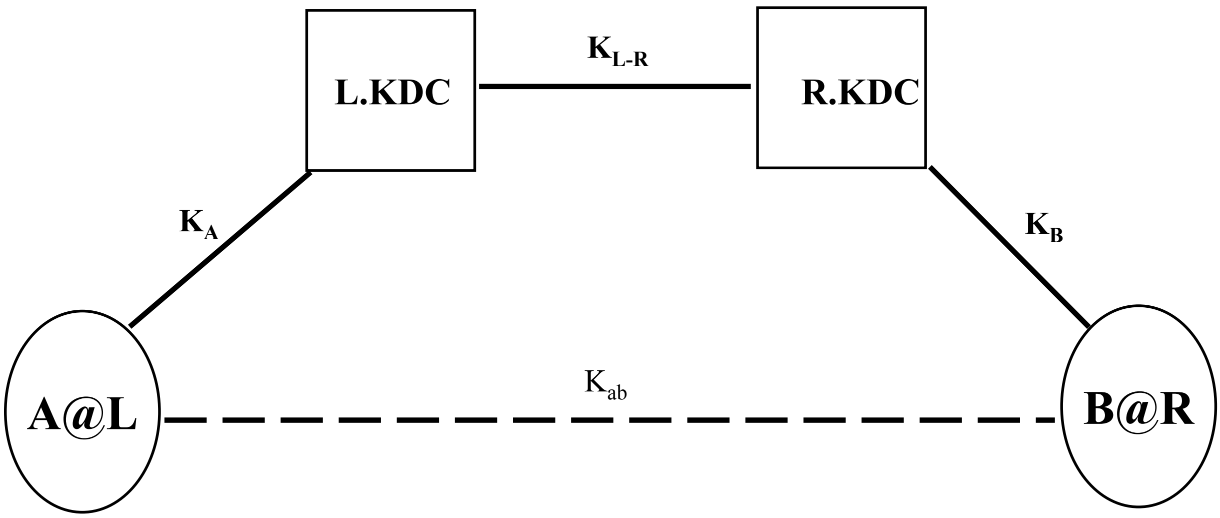
Threat: Unauthorized disclosure of users' passwords.
Unauthorized modification of user and account data
- create / modify user accounts and their properties;
- replace (encrypted) user's password entry with attacker's

Protection: Maintain the integrity of the Master KDC file copy in transit.
- compute a hash function of the Master KDC file copy.
- send the hash function to each Slave KDC in a krb_safe message.

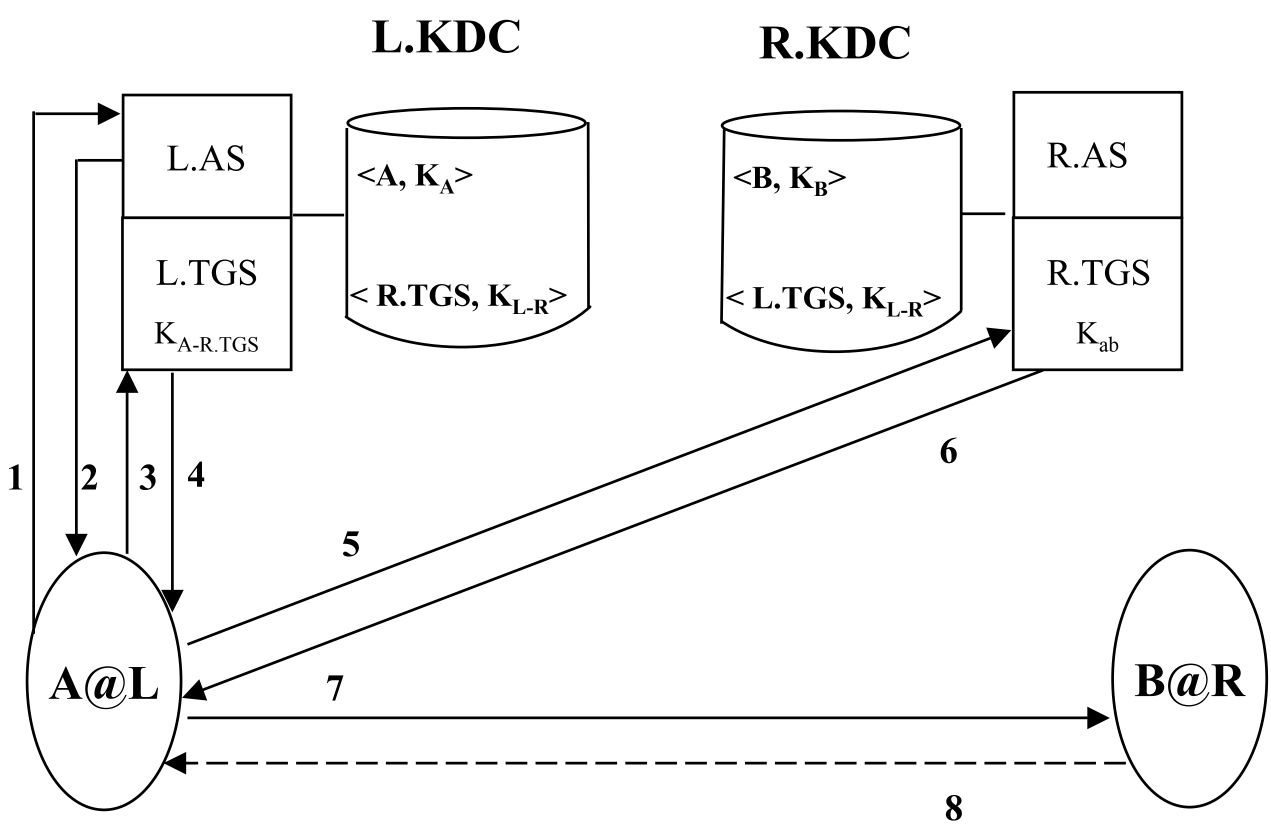
Residual Threat:
Ciphertext-only attack against the users' password entries.
Some user privacy concerns (e.g., user registration attributes).

Interrealm Authentication

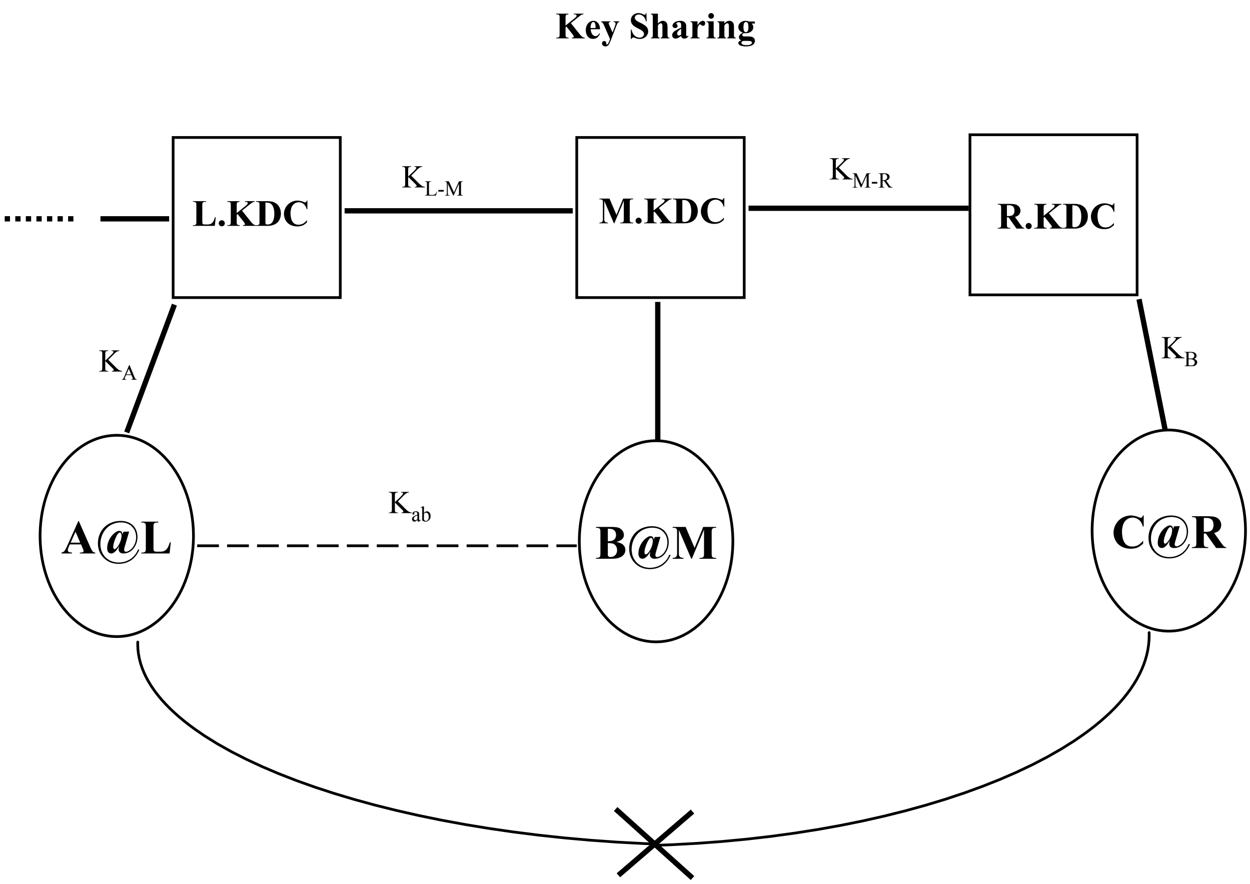
Key Sharing



Protocol Message Flows



Non-Transitive Authentication Trust (krb v4)



Motivation: Penultimate, rogue KDC of a KDC chain (i.e., L.KDC) can impersonate both local and foreign users.

Protection: User A.L's ticket for R.KDC (i.e., $K_{A\text{-}R.TGS}$) includes realm name L, and is made by M.KDC (i.e., encrypted with $K_{M\text{-}R}$).
Realm R.KDC will refuse a ticket made by M.KDC for a foreign user (i.e., a user of L.KDC, or of any other realm but M.KDC).

Limitation: Manage and protect O(n^2) shared cross-realm keys.
Establish O(n^2) trust relations.

Encryption for Confidentiality and Integrity

- **CBC Encryption Mode**

  Encryption and Decryption

- **IV Requirements**
- **CBC Invariant Property**

- **PCBC Encryption Mode**

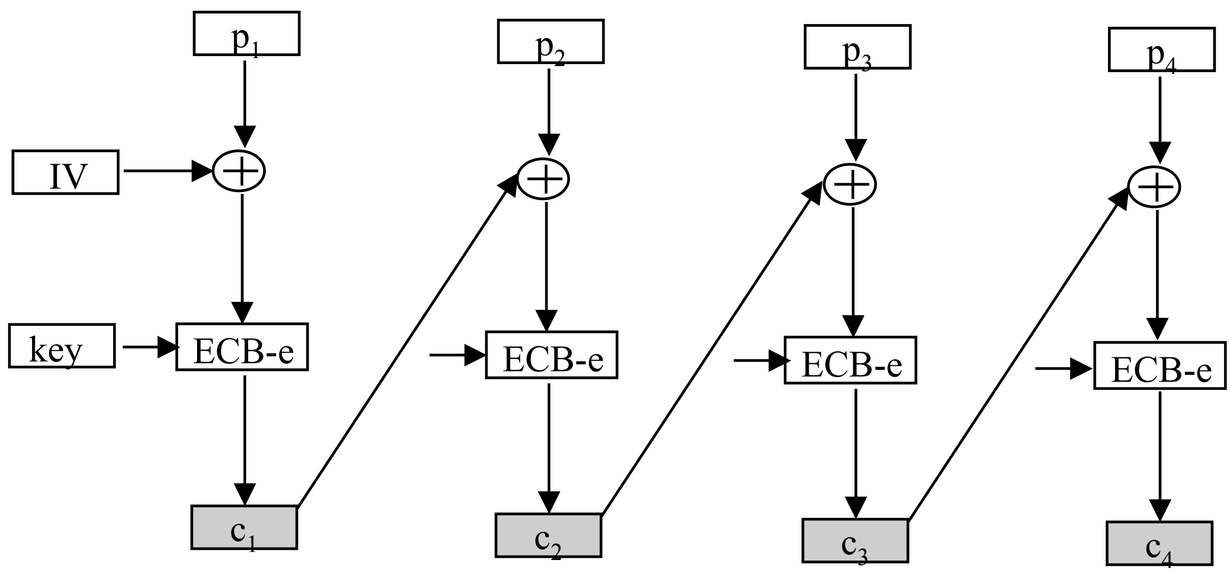
  Encryption and Decryption

- **PCBC Invariant Property**

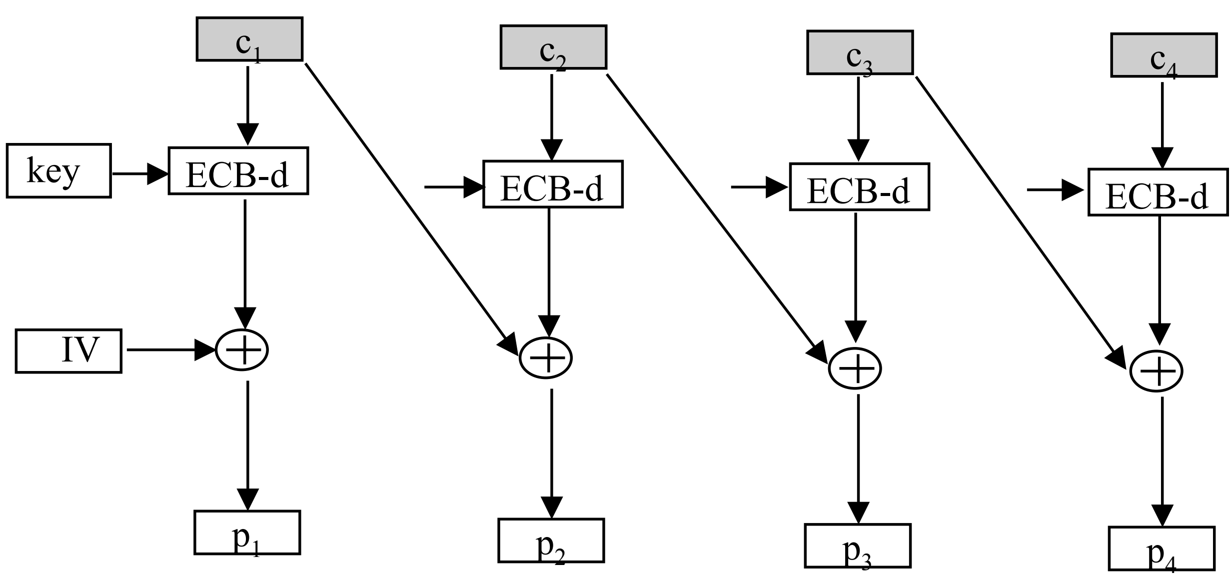
- **Data Encryption (for Confidentiality)**
- **Data Integrity**

CBC Encryption Mode

Encryption : $C_n = \{ C_{n-1} \oplus P_n \}^K$, where $C_0 = IV$



Decryption : $C_{n-1} \oplus \{C_n\}^{K^{-1}} = P_n$, where $C_0 = IV$



CBC Encryption Mode (ctnd)

IV Requirements

1. IV Must be Secret (and Random)

Chosen Plaintext Attack: Let $\mathbf{IV_a}$, $\mathbf{IV_b}$ be *known*, and **K**, $\mathbf{P_1}$ be *secret*.

Choose $\mathbf{X_i}$ such that $\{ IV_a \oplus X_i \}^K = \{ IV_b \oplus P_1 \}^K$

Then, $IV_a \oplus IV_b \oplus X_i = P_1$

2. IV Must be Selected / Changed per Association (e.g., per session)

Chosen Plaintext Attack: Let **IV** be *constant* (but *secret*) and $\mathbf{P_1}$ be *secret* (but *predictable*). $\mathbf{P_1}$ has a few known values P^1_1 , P^2_1 , ..., P^n_1

Steal $\{ IV \oplus P_1 \}^K$ and construct a table of 2^{56} entries for each P^i_1 , each entry containing $\{ IV \oplus P^i_1\}^{Kj}$

Find an entry s.t. $\{ IV \oplus P_1 \}^K = \{ IV \oplus P^i_1\}^{Kj}$ and (*secret*) key K = Kj.

3. IV Must be Protected from Predictable Modification

Modification Attack: Predictable change of **IV[i]** bit causes predictable change of $\mathbf{P_1[i]}$ bit, even if $\mathbf{P_1}$ is *secret*.

$$C_1 = \{ IV \oplus P_1 \}^K \Rightarrow \overline{P_1[i]} = \overline{IV[i] \oplus \{ C_1 \}^{K^{-1}} [i]} = \overline{IV[i]} \oplus \{ C_1 \}^{K^{-1}} [i]$$

CBC Encryption Mode (ctnd)

Encryption : $C_n = \{ C_{n-1} \oplus P_n \}^K$

Decryption : $C_n \oplus \{C_{n+1}\}^{K^{-1}} = P_{n+1} \Rightarrow$ *modify* $P_{n+1}[i]$

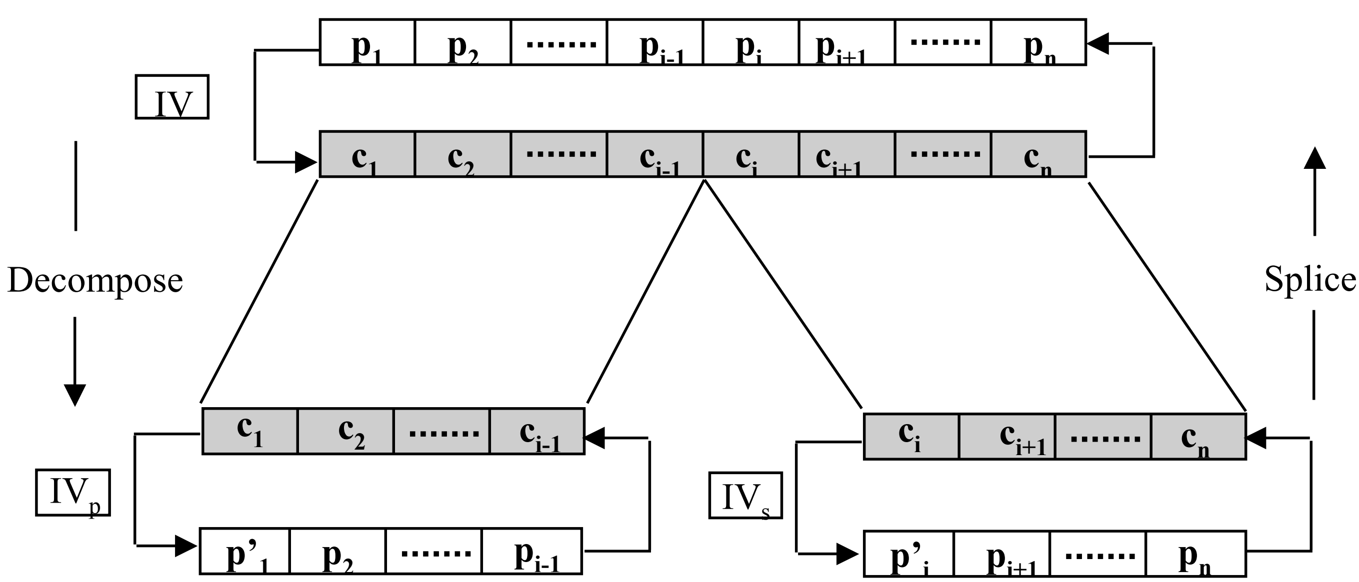
↑

modify $C_n[i]$

↓

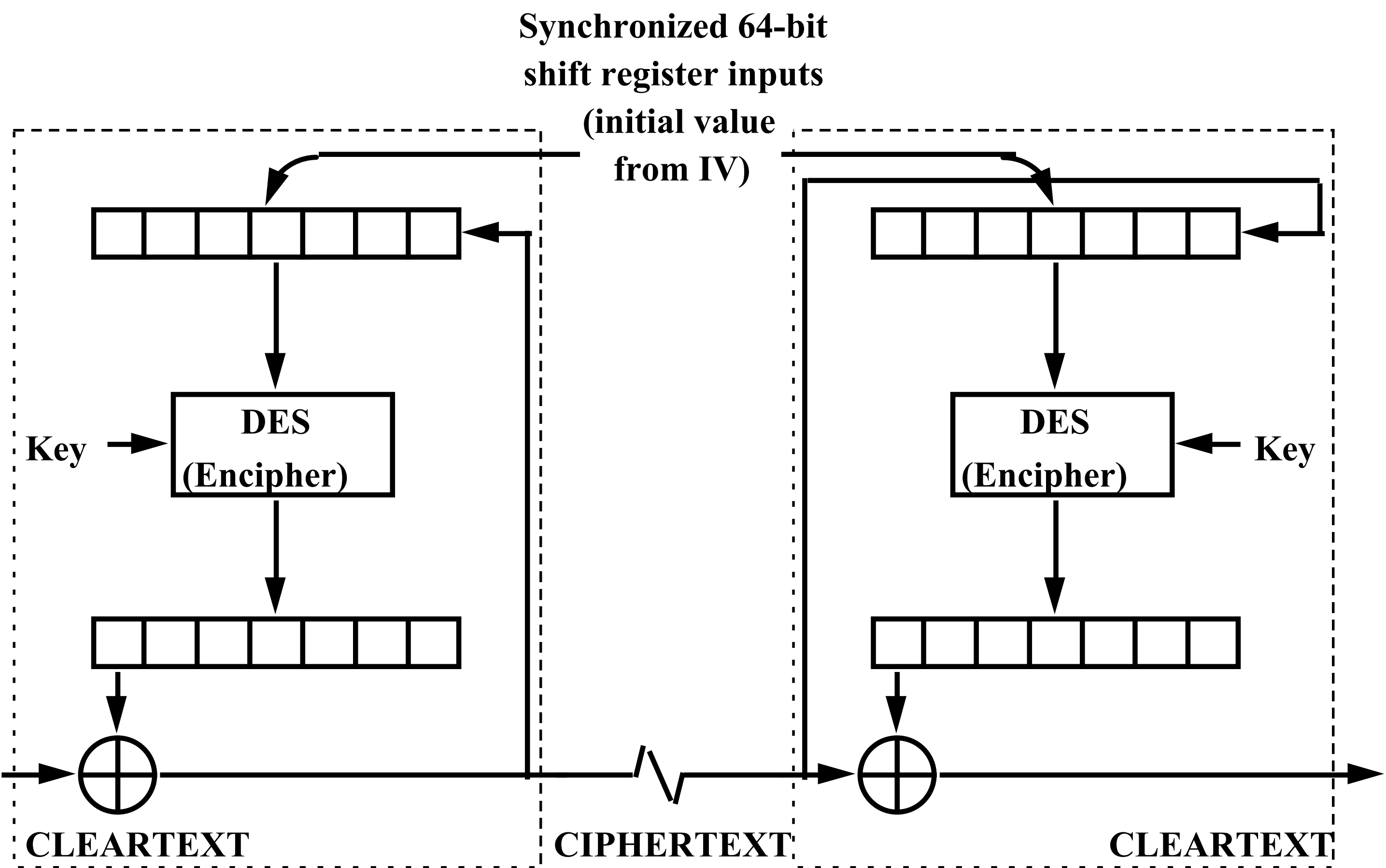
$C_{n-1} \oplus \{C_n\}^{K^{-1}} = P_n \Rightarrow$ *random* P_n

CBC Invariant Property



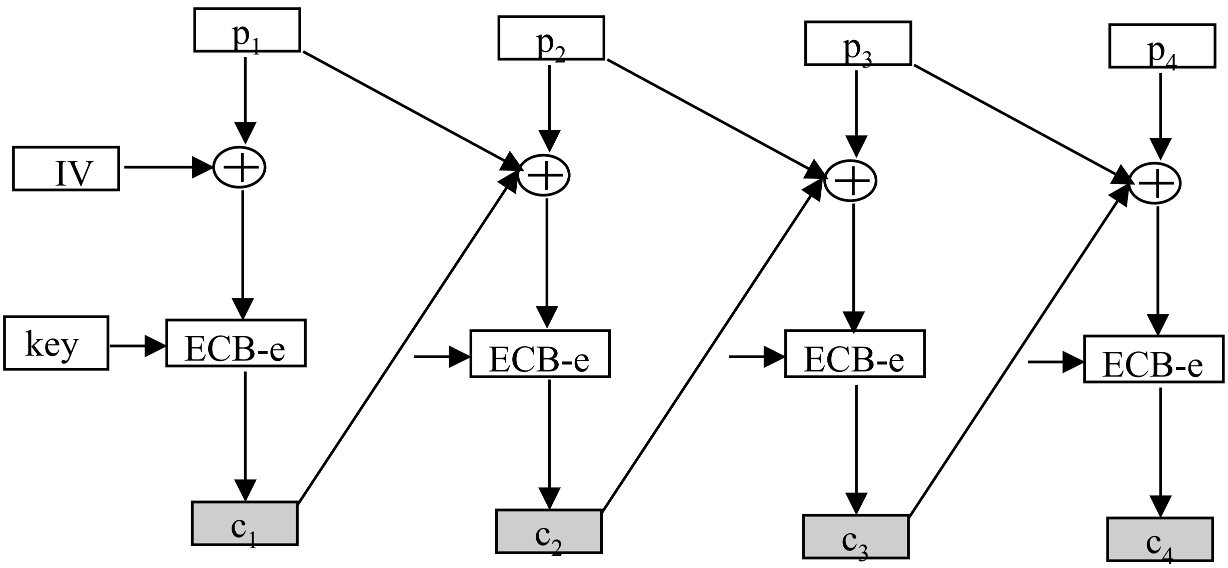
$P'_1 = P_1 \oplus IV_p \oplus IV$ $\qquad$ $P'_i = C_{i-1} \oplus P_i \oplus IV_s$

The Cipher Feedback (CFB) mode of the DES

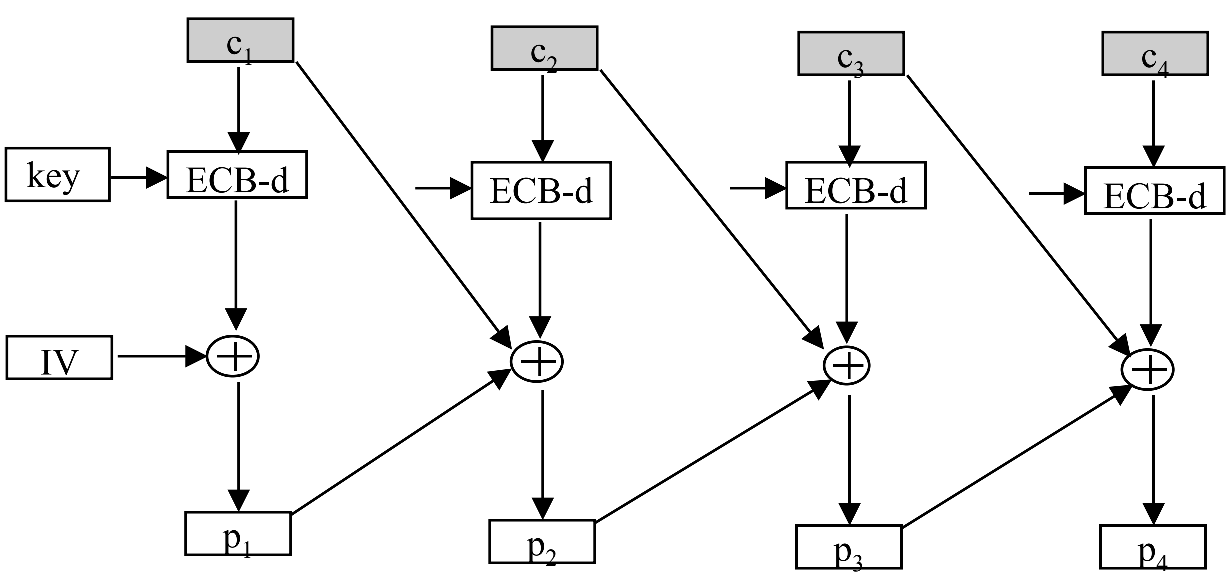


PCBC Encryption Mode

Encryption : $C_n = \{C_{n-1} \oplus P_{n-1} \oplus P_n\}^K$, where C_0=IV, P_0= 0



Decryption : $C_{n-1} \oplus P_{n-1} \oplus \{C_n\}^{K^{-1}} = P_n$, where C_0=IV, P_0= 0



PCBC Encryption Mode (ctnd)

Encryption : $C_n = \{C_{n-1} \oplus P_{n-1} \oplus P_n\}^K$

Decryption : $\{C_n\}^{K^{-1}} \oplus C_{n-1} \oplus P_{n-1} = P_n \Rightarrow$ *random* P_n

↑

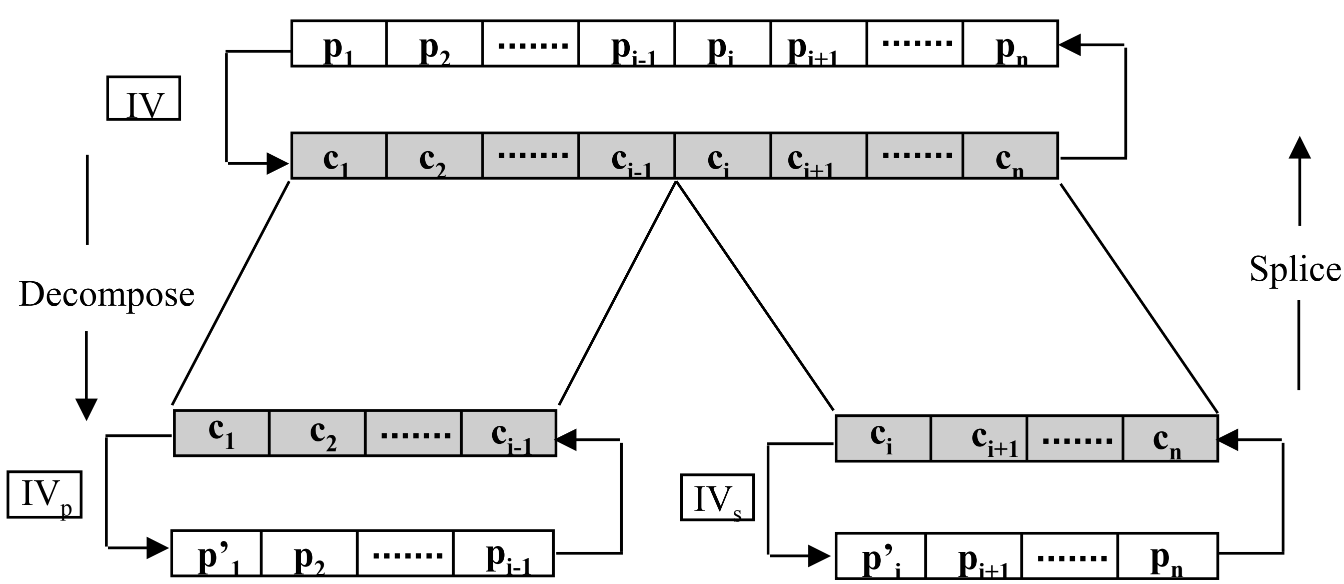
modify $C_n[i]$

↓ ↘

$C_n \oplus P_n \oplus \{C_{n+1}\}^{K^{-1}} = P_{n+1} \Rightarrow$ *random* P_{n+1}

.... $\Rightarrow$ *random* P_{n+m}

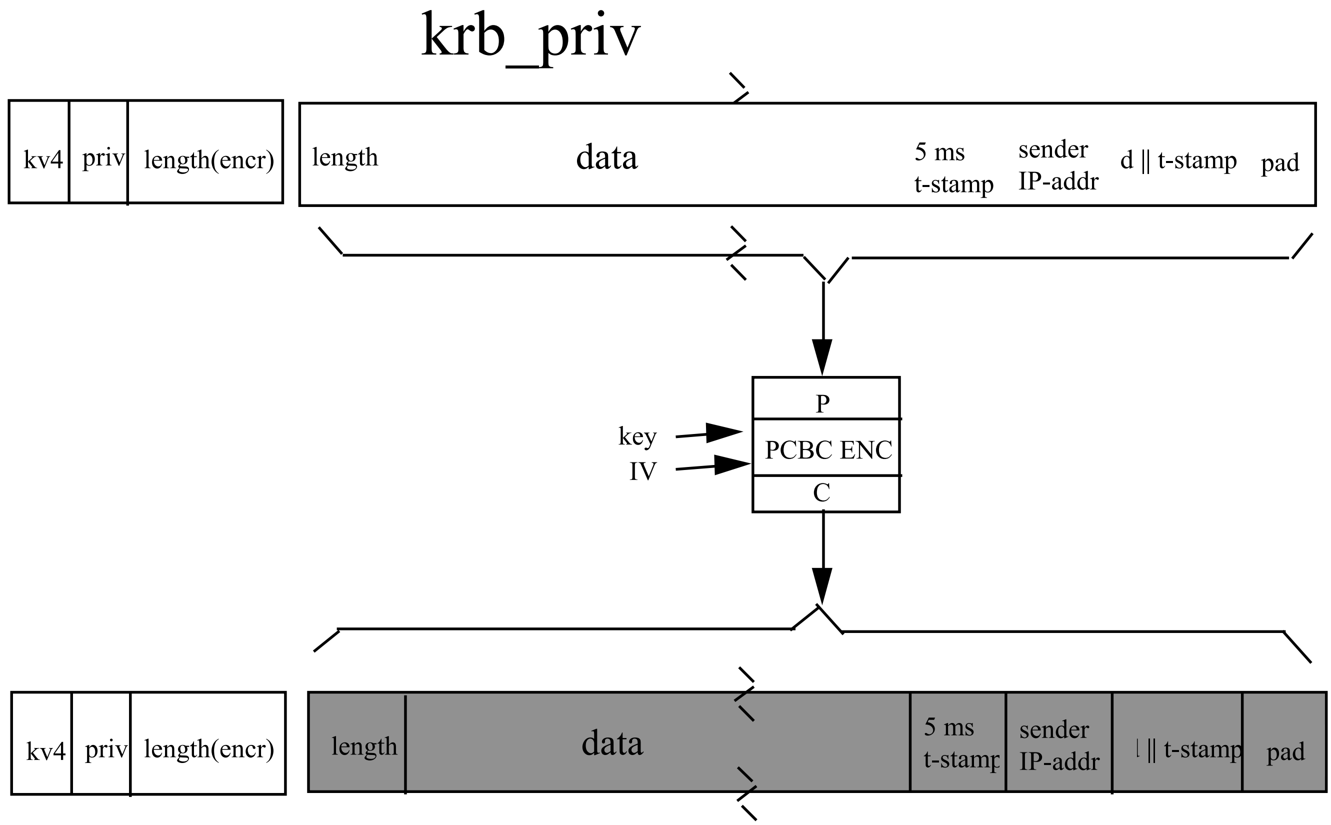
PCBC Invariant Property



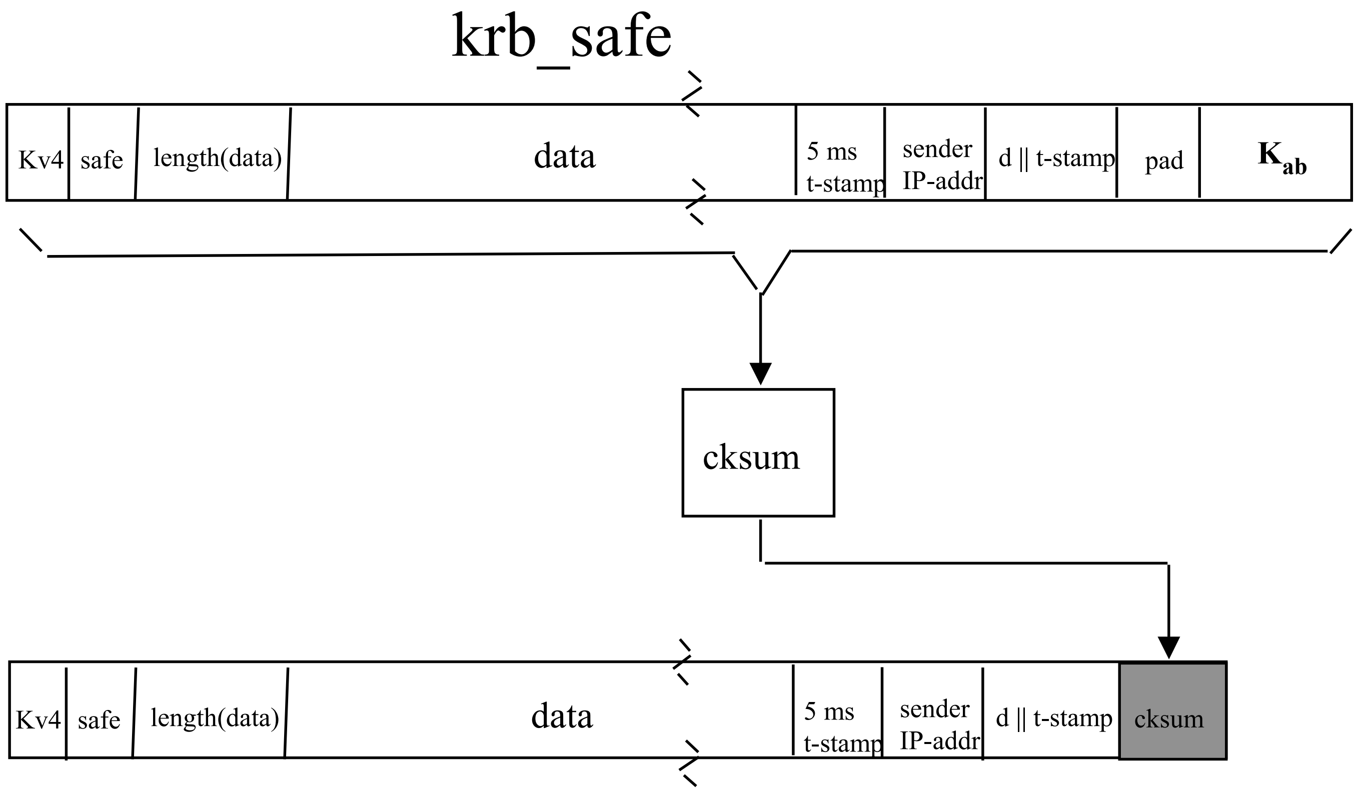
$P'_1 = P_1 \oplus IV_p \oplus IV$

$P'_i = C_{i-1} \oplus P_i \oplus P_{i-1} \oplus IV_s$

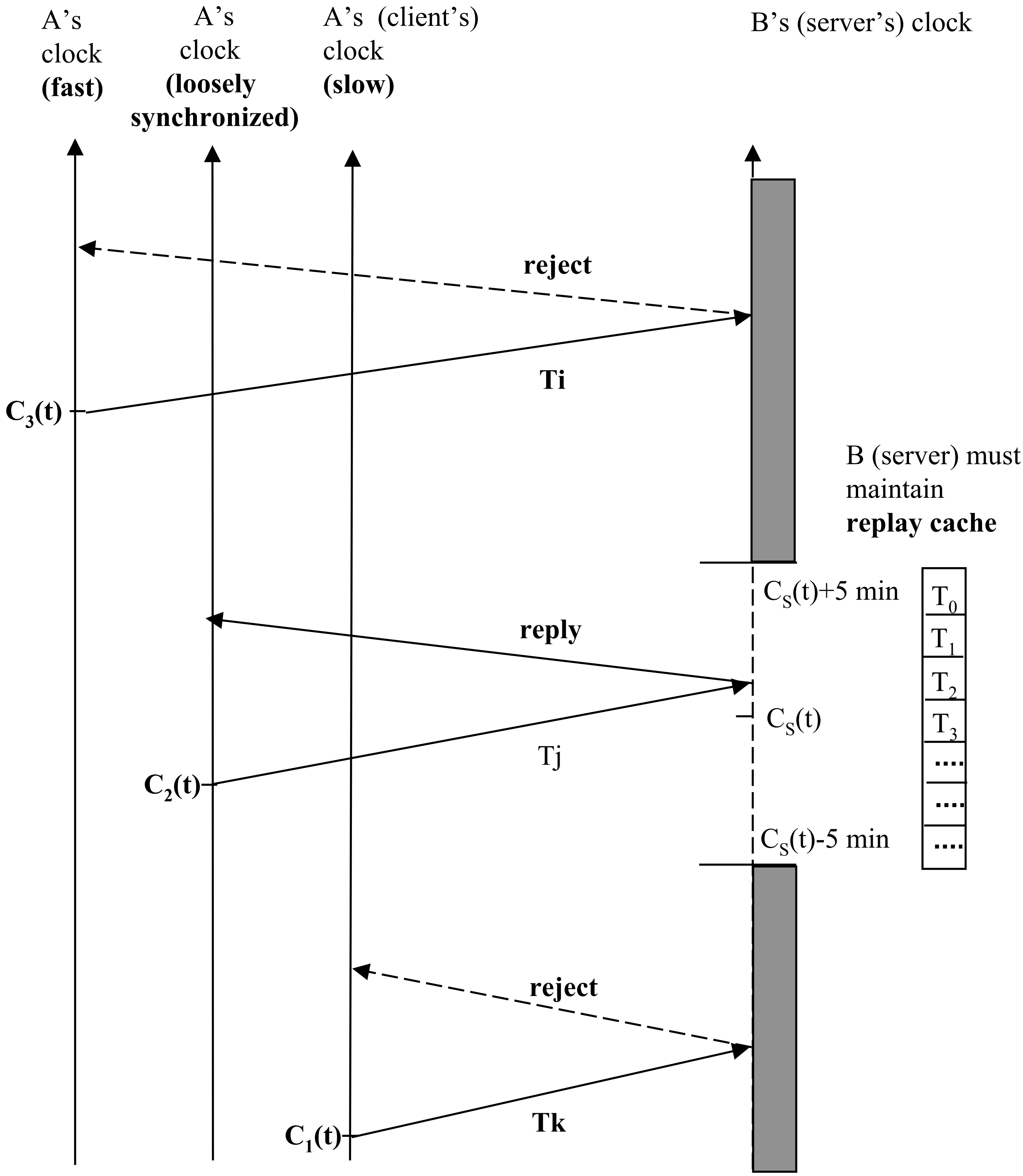
Data Encryption (for Confidentiality)



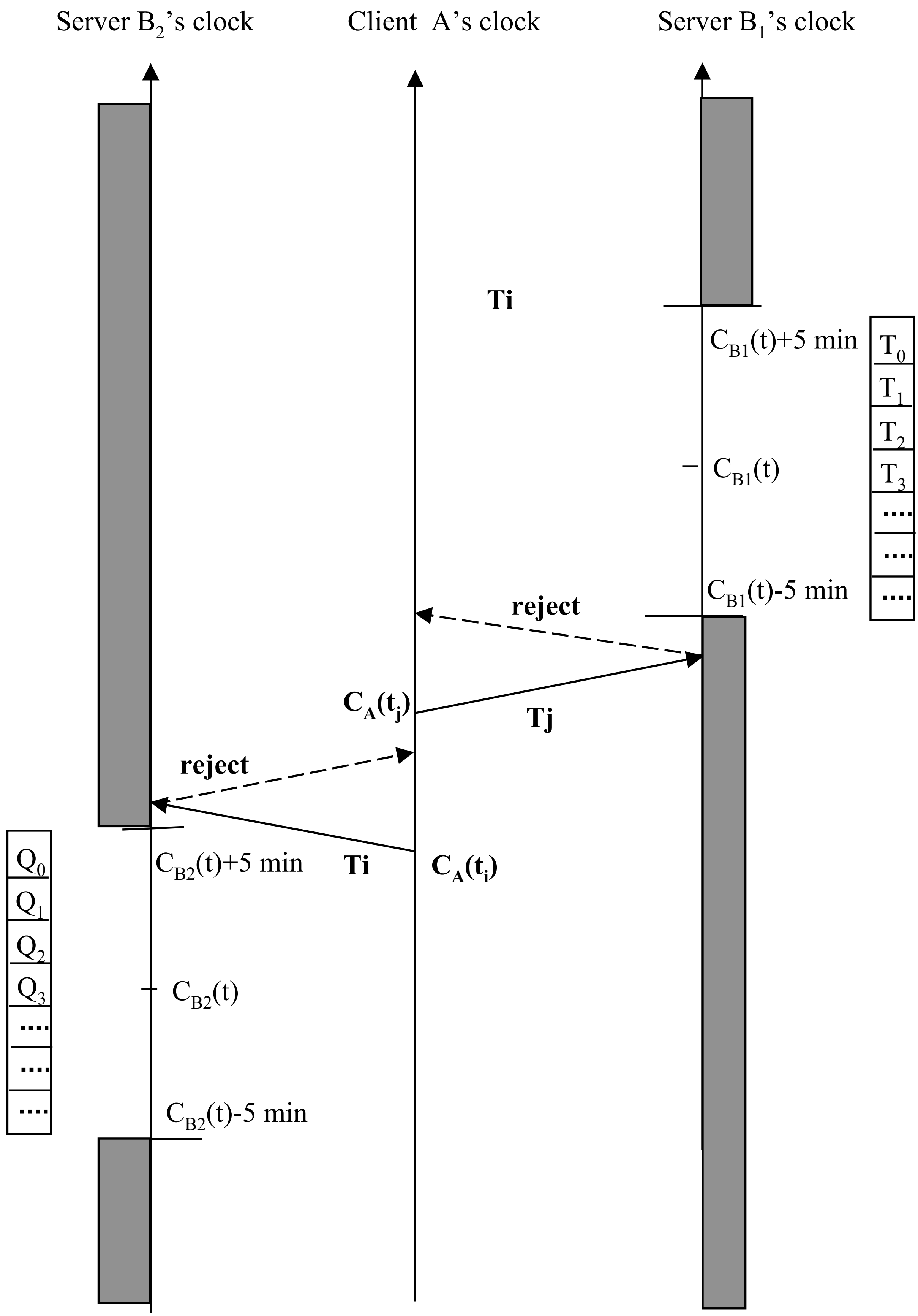
Data Integrity



Kerberos V4 Replay Detection
(sliding time window w/o server replay cache)



Out-of-Synch Clocks



Kerberos V4

Message Formats

Ticket

# bytes		
1	B	
≤40	A’s (i.e., client’s) name	null-terminated
≤40	A’s (i.e., client’s) instance	null-terminated
≤40	A’s (i.e., client’s) realm	null-terminated
4	A’s network-layer (e.g., IP) adddress	
8	session key for A <-> B (i.e., K_{ab})	
1	ticket lifetime (5 min. units)	
4	KDC timestamp (i.e., ticket issue time)	
≤40	B’s (i.e., server’s) name	null-terminated
≤40	B’s (i.e., server’s) instance	null-terminated
≤ 7	pad of 0’s to make ticket length a multiple of 8 bytes	

Authenticator

# bytes		
≤40	A's (i.e., client's) name	null-terminated
≤40	A's (i.e., client's) instance	null-terminated
≤40	A's (i.e., client's) realm	null-terminated
4	checksum	
1	A's (i.e., client's) timestamp (5 millisec.)	
4	timestamp	
≤ 7	pad of 0's to make ticket length a multiple of 8 bytes	

Credential field of a AS_REP or TGS_REP

# bytes		
8	session key for A <-> B (i.e., K_{ab})	
≤40	B's (i.e., server's) name	null-terminated
≤40	B's (i.e., server's) instance	null-terminated
≤40	B's (i.e., server's) realm	null-terminated
4	A's network-layer (e.g., IP) adddress	
1	ticket lifetime	
1	B's (i.e., server's) key version number	
4	ticket length	
≤40	ticket	null-terminated
≤40	KDC timestamp (i.e., ticket issue time)	null-terminated
≤ 7	pad of 0's to make cred. length a multiple of 8 bytes	

AS_REQ

# bytes	Field	
1	Kerberos version (4)	
1	message type (1) | B	
≤40	A's (i.e., client's) name	null-terminated
≤40	A's (i.e., client's) instance	null-terminated
≤40	A's (i.e., client's) realm	null-terminated
4	A's (i.e., client's) timestamp	
1	requested ticket lifetime	
≤40	B's (i.e., server's) name	null-terminated
≤40	B's (i.e., server's) instance	null-terminated

TGS_REQ

# bytes	TGT	
1	Kerberos version (4)	
1	message type (3) | B	
1	KDC's key version number	
≤40	KDC's realm	null-terminated
1	length of TGT	
1	length of authenticator	
variable	TGT	
variable	authenticator	
1	A's (i.e., client's) timestamp	
	requested ticket lifetime	
≤40	B's (i.e., server's) name	null-terminated
≤40	B's (i.e., server's) instance	null-terminated

AS_REP and TGS_REP

# bytes		
1	Kerberos version (4)	
1	message type (2) | B	
≤40	A’s (i.e., client’s) name	null-terminated
≤40	A’s (i.e., client’s) instance	null-terminated
≤40	A’s (i.e., client’s) realm	null-terminated
4	A’s (i.e., client’s) timestamp	
1	number of tickets (1)	
4	ticket expiration time	
variable 1	A’s (i.e., client’s) key version number	
2	credential length	
	credential	

AP_REQ

# bytes			
1	Kerberos version (4)		
1	message type (8)	B	
1	B's (i.e., server's) key version number		
≤40	B's (i.e., server's) realm		null-terminated
1	length of ticket		
1	length of authenticator		
variable	ticket		
variable	authenticator		

AP_REP - optional

# bytes		
1	Kerberos version (4)	
1	message type (6)	B
4	length of encrypted material (4)	
4	A's authenticator's checksum + 1	

AP_ERR

# bytes			
1	Kerberos version (4)		
1	message type (8)	B	
1	error code		
≤40	error text (additional information)		null-terminated

KDC Error Reply

# bytes	Field		
1	Kerberos version (4)		
1	message type (32)	B	
≤40	A’s (i.e., client’s) name		null-terminated
≤40	A’s (i.e., client’s) instance		null-terminated
≤40	A’s (i.e., client’s) realm		null-terminated
4	A’s (i.e., client’s) timestamp		
4	error code		
≤40	error text (additional information)		null-terminated

KRB_PRIV

# bytes		
1		Kerberos version (4)
1		message type (6) B
4		length of encrypted material (e.g., data)
4		length of data
variable		data
1		A's (i.e., client's) timestamp (5 millisec.)
4		A's (i.e., client's) network-layer (i.e., IP) address
4	D	timestamp
variable		pad of 0's to make length a multiple of 8 bytes

KRB_SAFE

# bytes		
1	Kerberos version (4)	
1	message type (7)	B
4	length of data	
variable	data	
1	A’s (i.e., client’s) timestamp (5 millisec.)	
4	A’s (i.e., client’s) network-layer (i.e., IP) address	
4	D	timestamp
16	(pseudo-Jueneman) checksum	

Laboratory Notes

- **KDC Installation**