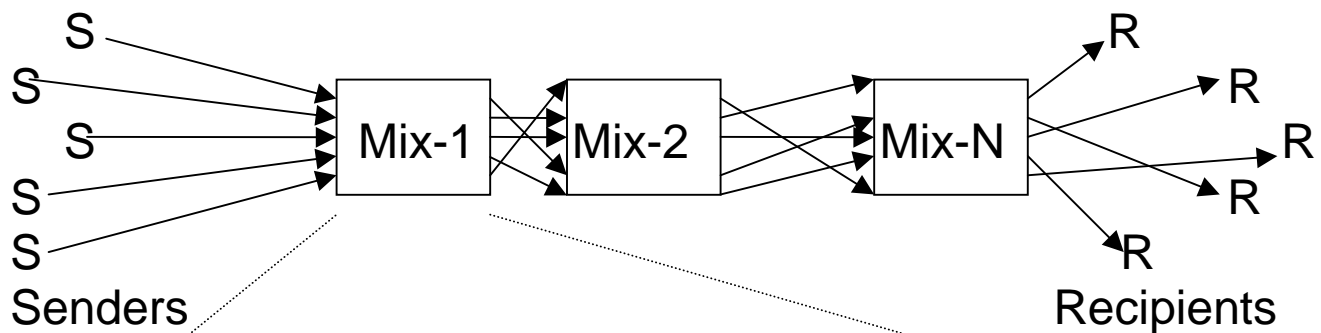


5. Mix Nets: Unlinkability of Sender and Recipient

5.1 Mix Concept [Chaum 1981]:



Mix-Functionality:

- discards repeated messages
- collects messages in a batch / pool
- changes the outlook,
(e.g. decrypts $M_1 = c_1(z_1, A_2, M_2)$ with d_1 , ignores z_1 , obtains address $A_2 +$ (encrypted) M_2)
- reorders output messages
- outputs M_2 to Mix-2 (with address A_2)

5.2 Sender Anonymity

Sender chooses Mix-Sequence $Mix_1, \dots, Mix_n, Mix_{n+1}$

Mix_{n+1} = recipient

A_i ($i=1..n+1$): address of Mix_i

c_i ($i=1..n+1$): public key of Mix_i

z_i : random bit strings

M : message for recipient

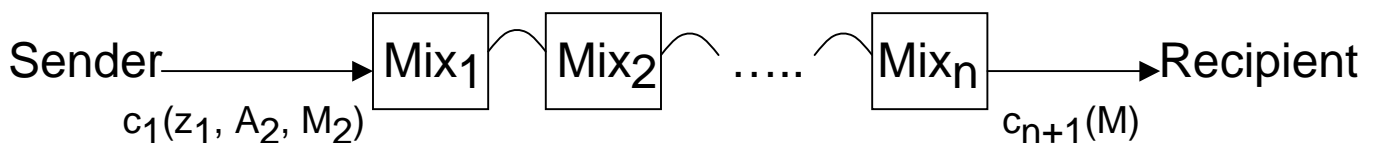
M_i : message that Mix_i will receive

Sender prepares his message:

$$M_{n+1} = c_{n+1}(M)$$

$$M_i = c_i(z_i, A_{i+1}, M_{i+1}) \quad \text{for } i=1\dots n$$

and sends M_1 to Mix_1



Each Mix_i decrypts:

$$c_i(z_i, A_{i+1}, M_{i+1}) \rightarrow$$

A_{i+1} : address of next Mix

M_{i+1} : $c_{i+1}(z_{i+1}, A_{i+2}, M_{i+2})$,
encoded message for Mix_{i+1}

z_i : random string,
to be discarded

and forwards M_{i+1} to Mix_{i+1}

5.3 Recipient Anonymity

Recipient B chooses Mix-Sequence Mix_1, \dots, Mix_m and creates anonymous return address RA:

$$R_{m+1} = e$$

$$R_j = c_j(k_j, A_{j+1}, R_{j+1}) \quad \text{for } j=1..m$$

$$RA = (k_0, A_1, R_1)$$

e : label of return address

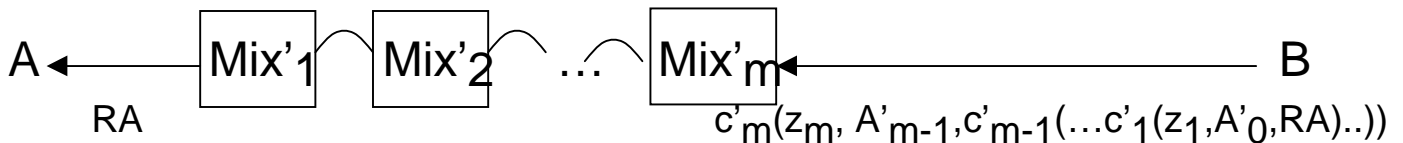
k_j : symmetric key, used by Mix_j to encode message

A_j ($j=1..m$): address of Mix_j

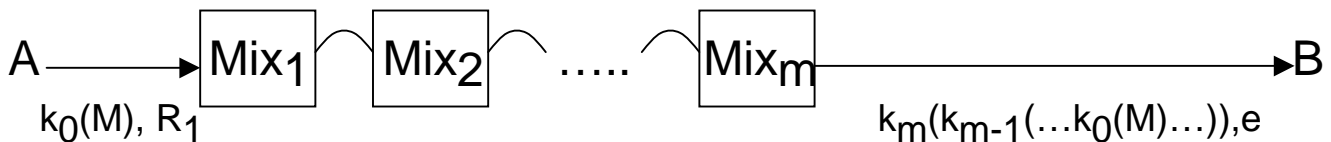
c_j ($j=1..m$): public key of Mix_j

z_j : random bit strings

Recipient B sends RA anonymously to Sender A:



Sender A replies:



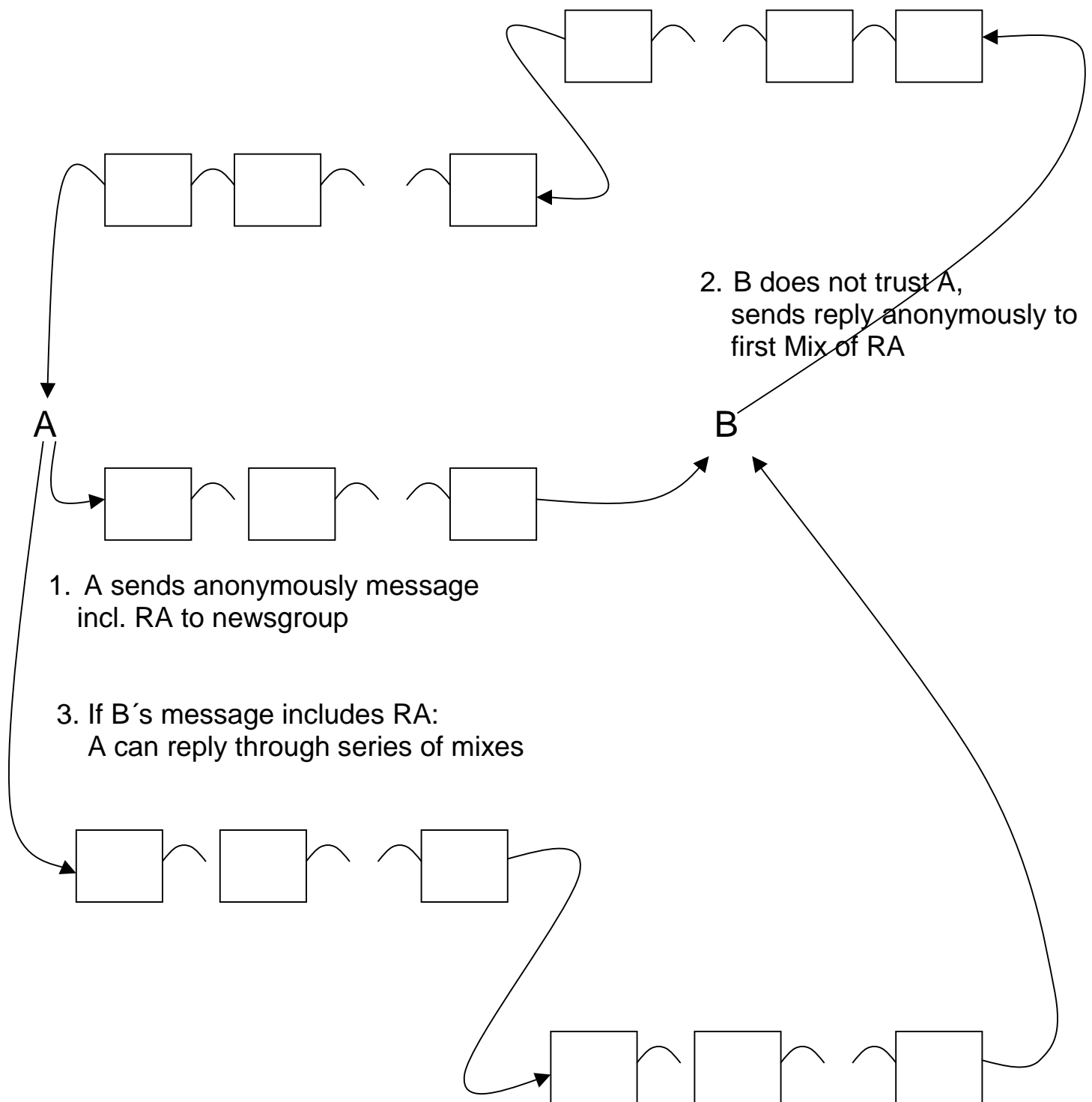
Each Mix_j receives: $k_{j-1}(\dots k_0(M) \dots), R_j$,

decrypts: $R_j = c_j(k_j, A_{j+1}, R_{j+1}) \rightarrow (k_j, A_{j+1}, R_{j+1})$,

forwards: $k_j(k_{j-1}(\dots k_0(M) \dots)), R_{j+1}$ to Mix_{j+1}

Label e indicates which k_0, \dots, k_m has to use to decrypt M

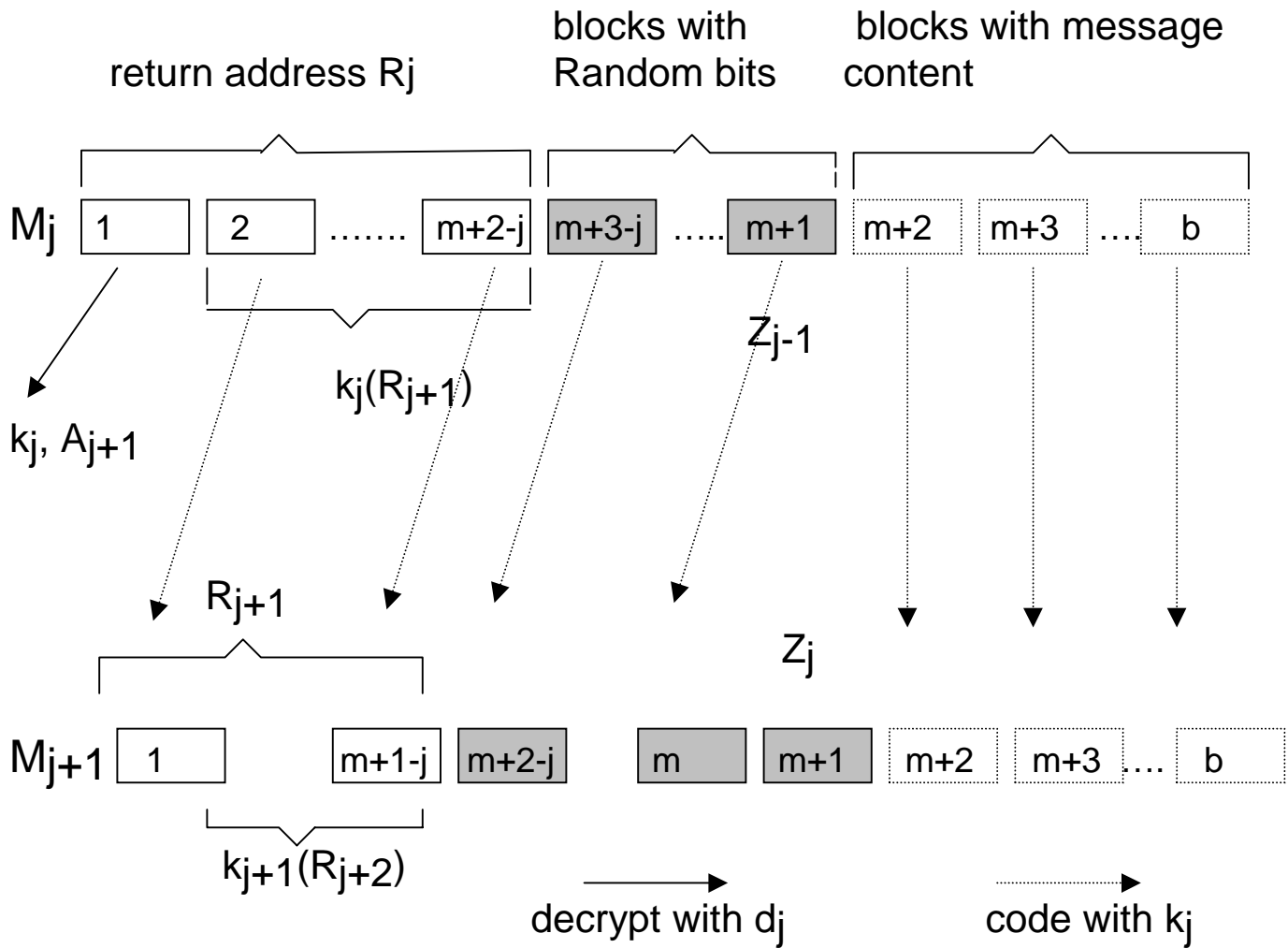
5.4 Two-Way Anonymous Conversation:



5.5 Length-Preserving Coding:

Aim: Prevent tracing of messages by (decreasing) size

- Messages are sent through Mix sequence Mix_1, \dots, Mix_m
- Each message has fixed length of b blocks



Creation of return address:

$$R_{m+1} = [e] \quad ([] = \text{block limits})$$

$$R_j = [c_j(k_j, A_{j+1})], k_j(R_{j+1}) \quad j=1, \dots, m$$

e : label , k_j : symmetric keys

Each Mix_j decrypts first block $c_j(k_j, A_{j+1}) \rightarrow k_j, A_{j+1}$,
 deletes first block, encrypts rest of M_j with K_j ,
 inserts Z_j before message blocks, forwards M_{j+1} to Mix_{j+1}

5.4.1 Length Preserving Coding providing Sender Anonymity

Recipient does not know symmetric keys k_1, \dots, k_m

→ Sender has to encrypt message with all k_i and to create R_1

Sender creates H_1MC_1 with (MC: message content)

$$H_1 = R_1$$

$$MC_1 = k_1(k_2 \dots (k_m(c_{m+1}(MC))))..)$$

c_{m+1} : public key of recipient

Each Mix_j *decrypts* message blocks with k_j

5.4.1 Length Preserving Coding providing Recipient Anonymity

Sender does not know symmetric keys k_1, \dots, k_m

Sender receives $RA = (k_0, A_1, R_1)$, encrypts MC with k_0 , and thus creates H_1MC_1 with

$$H_1 = R_1$$

$$MC_1 = k_0(MC)$$

Each Mix_j *encrypts* message blocks with k_j

5.5 Attacks & Contermeasures:

Passive attacks:

- Correlation by content :
 - ➔ all message to / from Mix should be encrypted and have to include random string

- Correlation by message length :
 - ➔ uniform message length (through padding)

- Time correlation :
 - ➔ *Output batch*: accumulate N messages, forward them in random order
 - ➔ *Pool*: If (N+1)th message arrives, forward one message from the pool
 - ➔ Combination of batch + pool
 - ➔ *Interval batching*: Fill batch/pool with dummy messages at end of time interval T
 - ➔ random delay

Active attacks:

- Isolate & Identify ((n-1)-attack):
 - Dummy messages
 - Check sender ID

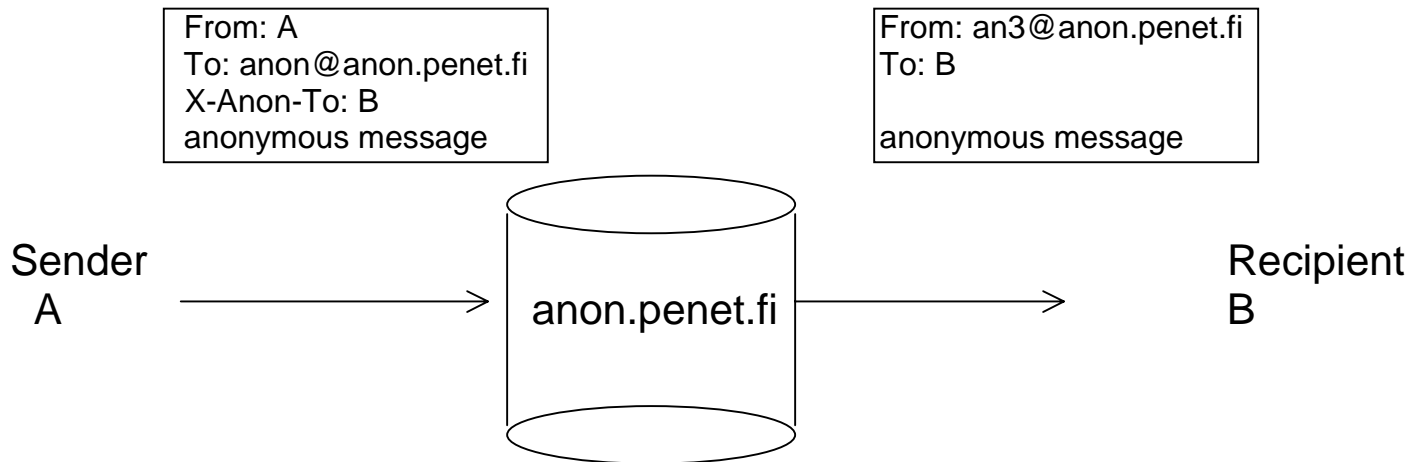
- Message replay attacks:
 - Discard replays
 - Replay detection through:
 - Log of sent messages (message IDs)
 - For hybrid encryption: Message ID = public key encrypted form of session key
 - Time stamps
 - Intermix detours
 - charge of Ecash

5.6 Applications of Mix-Concept for:

- Email (e.g., Cypherpunk Remailers, Mixmaster)
- ISDN
- Mobile Communication
- Interactive Internet Communication:
Web-Browsing, Remote Login, etc.
(e.g., Anonymizer, Pipenet, Onion Routing, Freedom network)

5.7 Applications for Email (Anonymous Remailers)

5.7.1 Anon Servers (Type-0 Remailers):



Advantage: easy to use

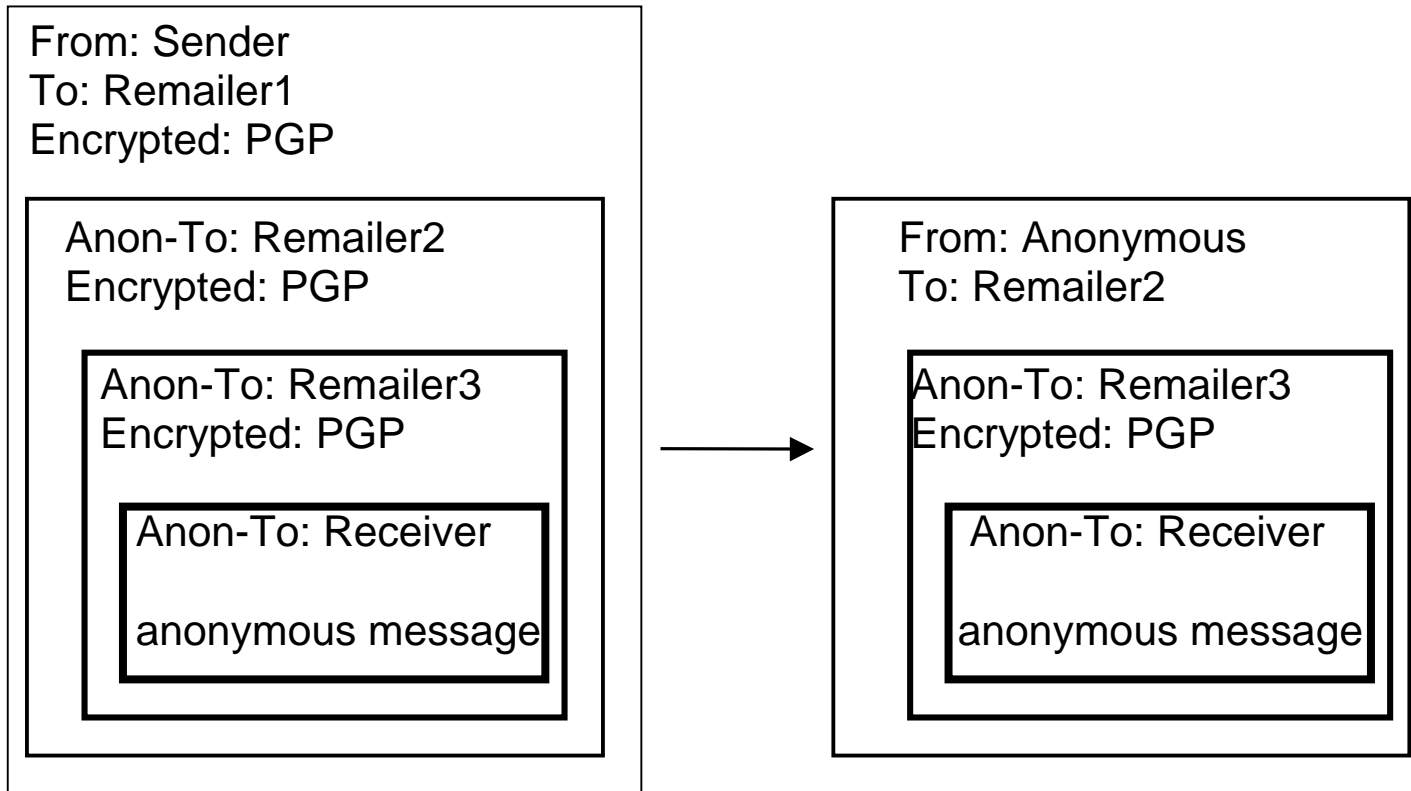
Drawbacks: easy to attack:

- no encryption -> messages could be intercepted
- anon-server has database with pseudo-ID -user-ID mappings

-> Helsingius' Anon-Server closed down since 1996
after intrusion by Finish Police /FBI

5.7.2 Cypherpunk Remailers:

Message sequence:



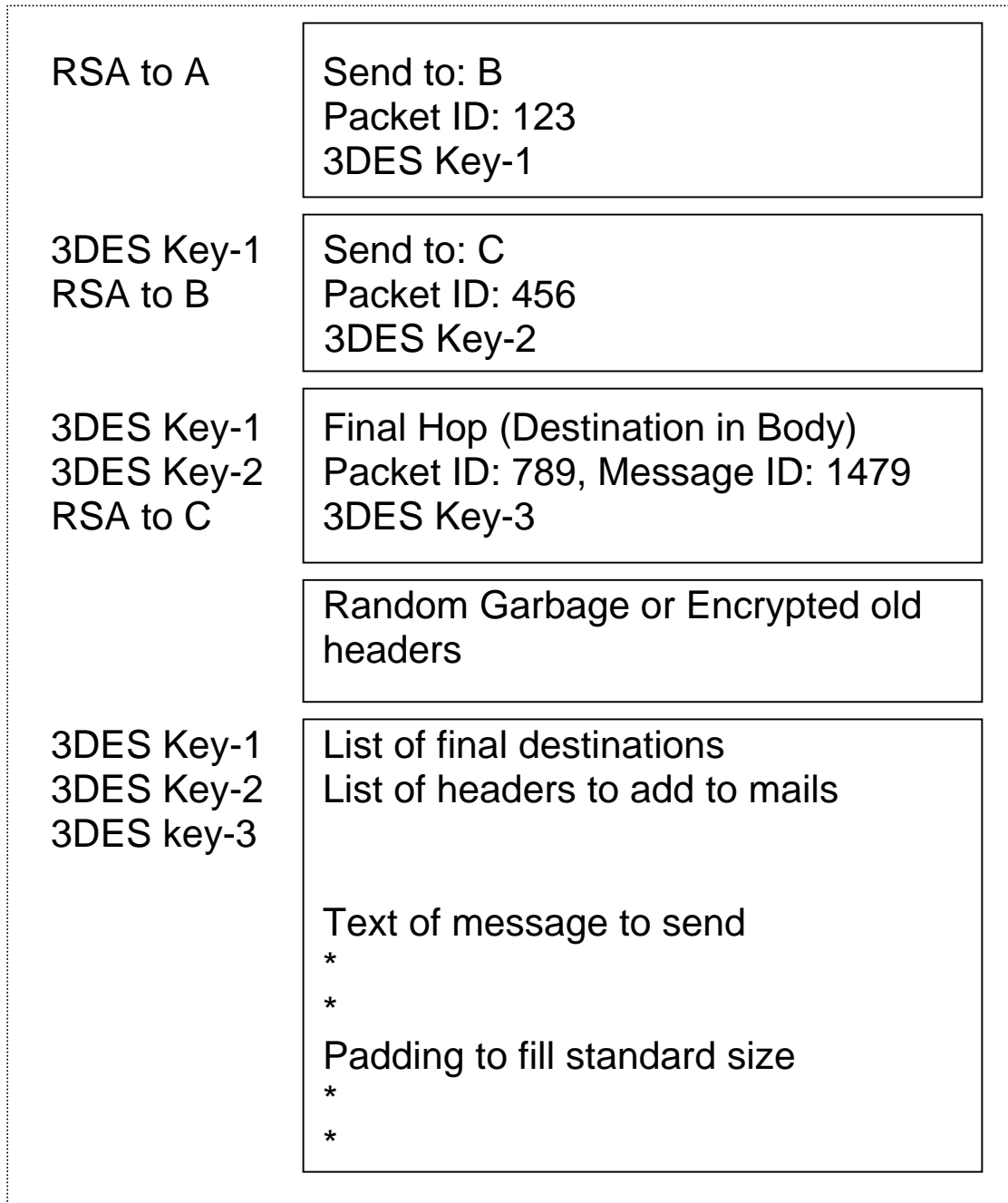
Possible Attacks:

- Tracing of messages, if incoming messages are directly forwarded
- Tracking of messages by size
- Replay attack

Countermeasures:

- ⇒ Latency, reordering
- ⇒ All messages should be exactly the same size
- ⇒ Remailers should refuse to send messages more than once

5.7.3 Mixmaster:



- Messages are sent as one or more packets
(of same size: 20 headers of 512 bytes, 10kB body)
- Repeated messages are discarded
- Messages are reordered
- Dummy messages

5.8 Applications for Interactive Internet-Communication

5.8.1 Anonymizer (<http://www.anonymizer.com>)

Web-proxy that filters out identifying headers and source addresses from Web browser



Functionalities:

Anonymisation of data stream:

- it does not forward the source IP address of the end-user
- it eliminates revealing information about the user's machine, the user's name, the previously-visited site name
- it does not forward the user's email address to serve as a password for FTP transactions;
- it filters out Java applets, JavaScript scripts, cookies

Drawbacks:

- No chaining of mixes -> Anonymizer is single point of trust
- Connection itself is not anonymised

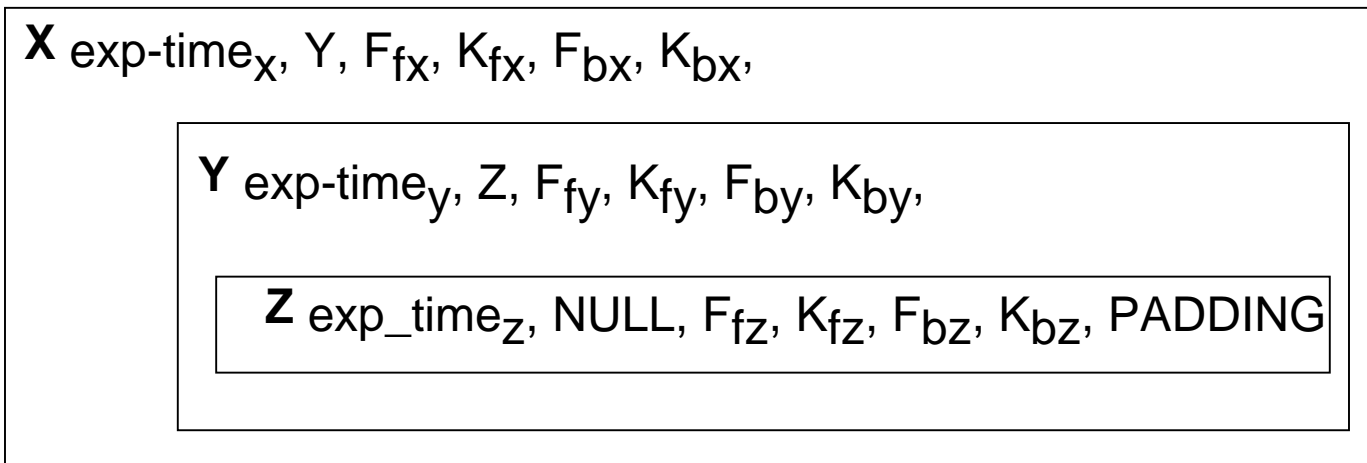
5.8.1 Onion Routing

Onion Routing provides anonymous socket connections by means of proxy servers

Onion = layered object to produce anonymous bi-directional virtual circuit between communication partners

Initiator's proxy constructs a "forward onion" which encapsulates a series of routing notes forming a route to the responder

Example: Forward Onion for the route W-X-Y-Z:



Each node receives:

{exp-time, next-hop, F_f, K_f, F_b, K_b, payload} PK_x

exp-time: expiration time

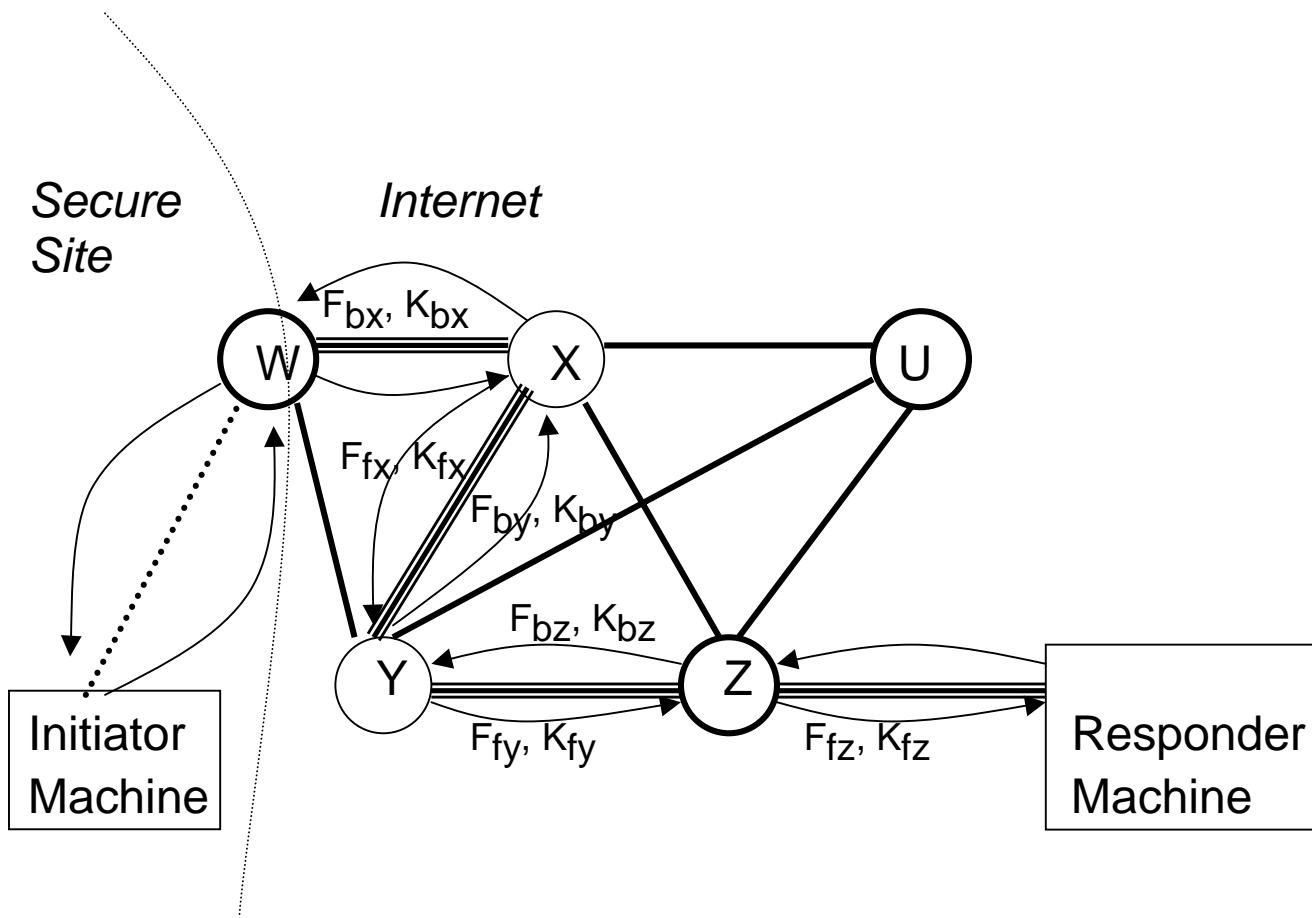
next_hop: next routing node

(F_f, K_f) : function / key pair for encrypting data moving forward in the virtual circuit

(F_b, K_b) : function/key pair for encrypting data moving backwards in the virtual circuit

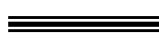
payload: another onion (or null for responder's proxy)

Example: Virtual Circuit



 : Data Flow (with Function / Key if crypted)

 : Unsecured Socket connection

 : Virtual circuit through link-encrypted connections between routing nodes

 : Link encrypted connections between routing nodes

 : Routing Node

 : Routing/ Proxy Node

Reply Onion: allows for a responder to send back (anonymously) a reply after the original circuit is broken

Example: A Reply Onion:

Z exp-time_z, Y, F_{bz}, K_{bz}, F_{fz}, K_{fz},

Y exp-time_y, X, F_{by}, K_{by}, F_{fy}, K_{fy},

X exp_time_x, W, F_{bx}, K_{bx}, F_{fx}, K_{fx},

W exp_time_x, NULL, NULL, NULL, NULL,
NULL, {IDENTITY, F_{bx}, K_{bx}, F_{fx}, K_{fx}, F_{by},
K_{by}, F_{fy}, K_{fy}, F_{bz}, K_{bz}, F_{fz}, K_{fz}, PADDING}

5.9 Possible Misuse of Mix-Technology:

- Illegal Forwarding of copyright-protected material / espionage information etc.
- Distribution of illegal material, junk mails, logic bombs
- Denial of Service attacks
- Hacking without traces