Symmetric Encryption in Automatic Analyses for Confidentiality against Active Adversaries

Peeter Laud
Tartu University & Cybernetica AS
http://www.ut.ee/~peeter_l

Problem statement

- Given a cryptographic protocol
  - More generally, a distributed computing system
- It works with some secret data
- No outside adversary should be able to learn anything about this secret data
  - Even when allowing active attacks

Problem statement (contd.)

- We fix a programming language
- ... and its semantics
- The specification of the system is given
  - Each part is implemented in that language
- We must decide, whether it is secure
  - Automatically
  - Which is not always possible (problem undecidable)
  - Err to the safe side

Running example

- Transmit the secret M from A to B:

$$
\begin{array}{ll}
A \rightarrow S: & enc(K_{AS}: B, K_{AB}) \\
S \rightarrow B: & enc(K_{BS}: A, K_{AB}) \\
A \rightarrow B: & enc(K_{AB}: M) \\
B \rightarrow \ : & \text{OK}
\end{array}
$$

- S is a server, trusted by A and B
- K_{AS} and K_{BS} are long-term keys shared by S and A resp. B

The semantics

- We don't use Dolev-Yao semantics / intruder
- All values are bit-strings
  - Tagged by their type
- Operations are implemented by probabilistic polynomial-time (PPT) algorithms
- The adversary may be any PPT algorithm
  - ... it does not have to tag the values correctly

Running example

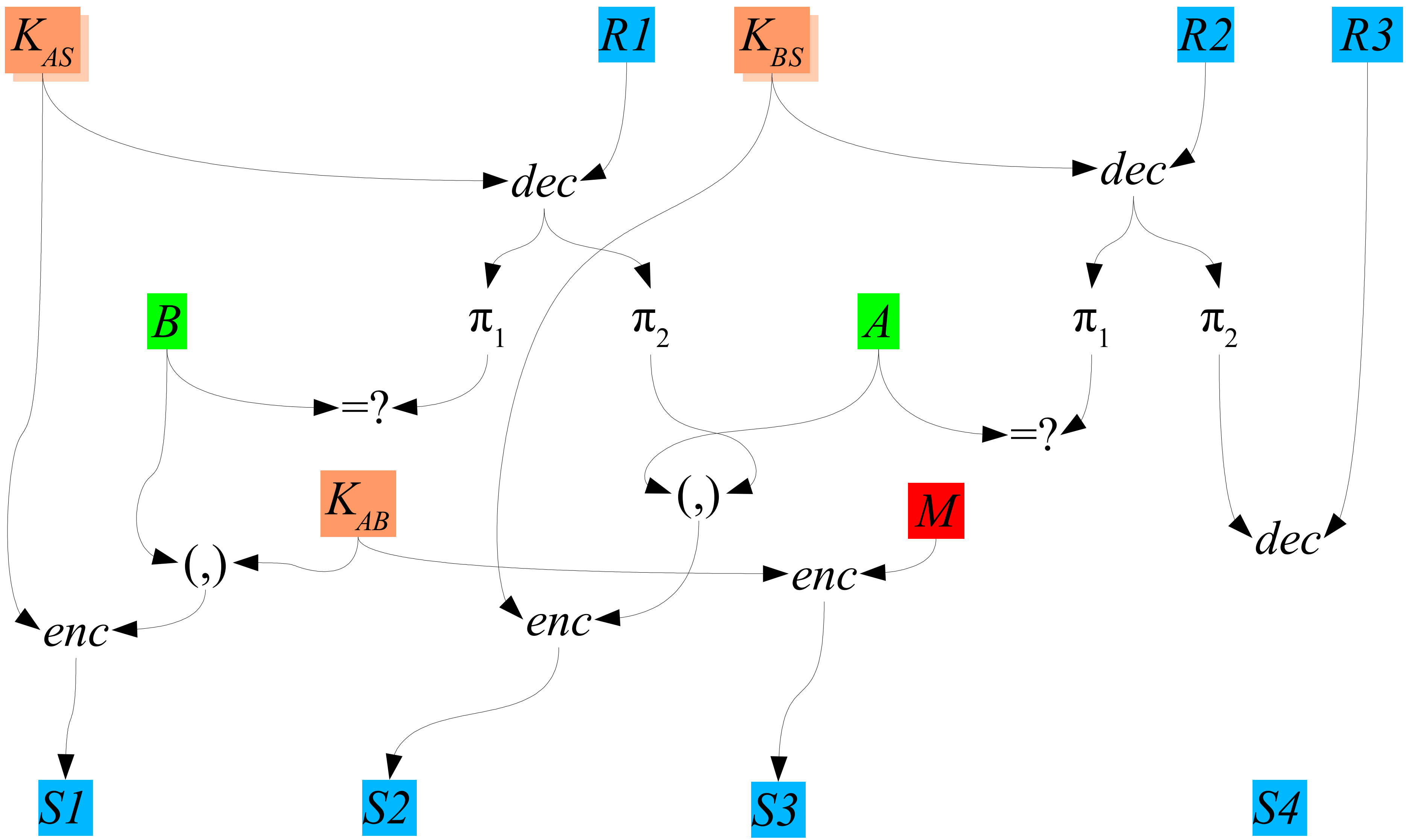
$$A \rightarrow S: \quad enc(K_{AS}: B, K_{AB})$$

$$S \rightarrow B: \quad enc(K_{BS}: A, K_{AB})$$

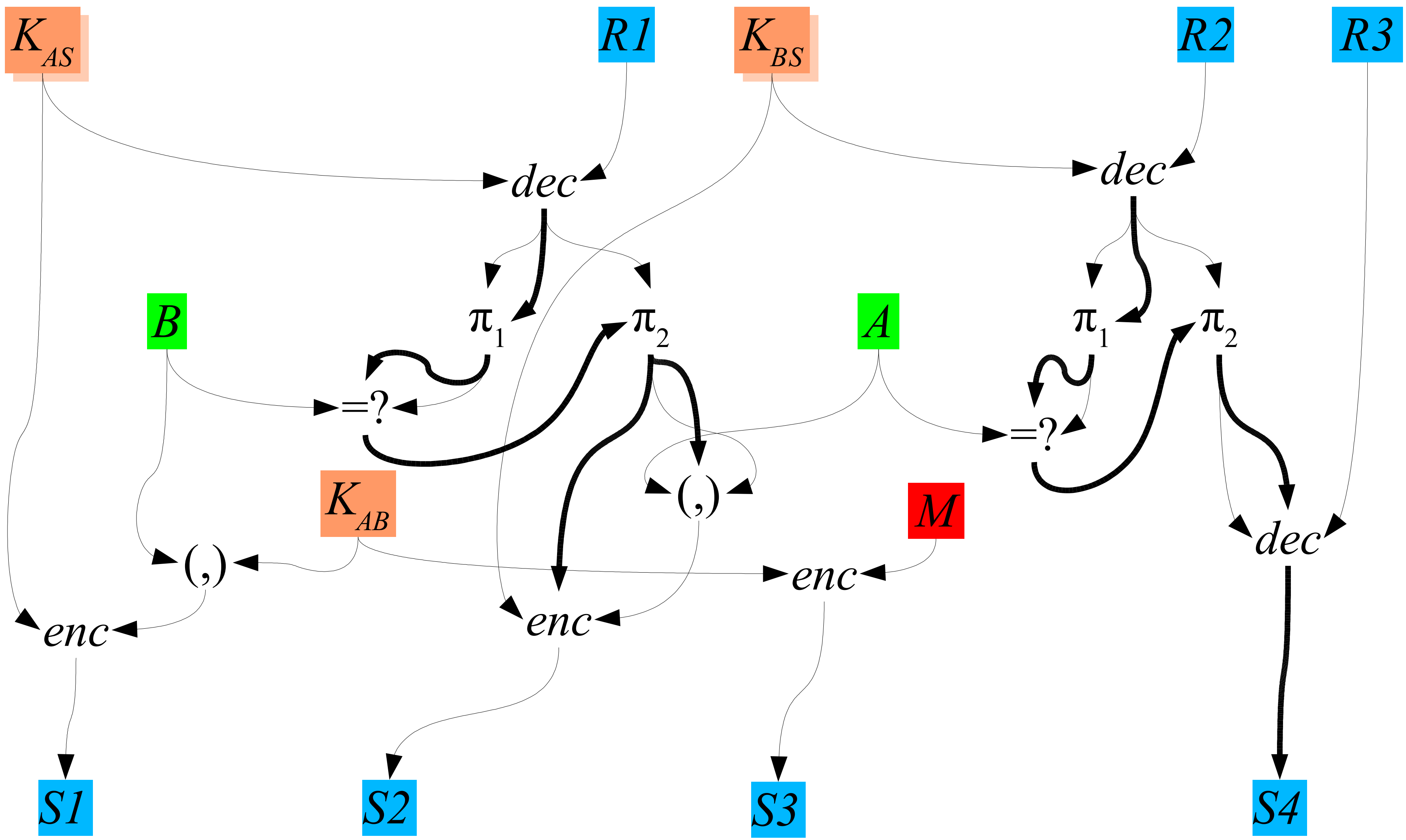
$$A \rightarrow B: \quad enc(K_{AB}: M)$$

$$B \rightarrow \ : \quad \text{OK}$$

Data dependencies



Control dependencies



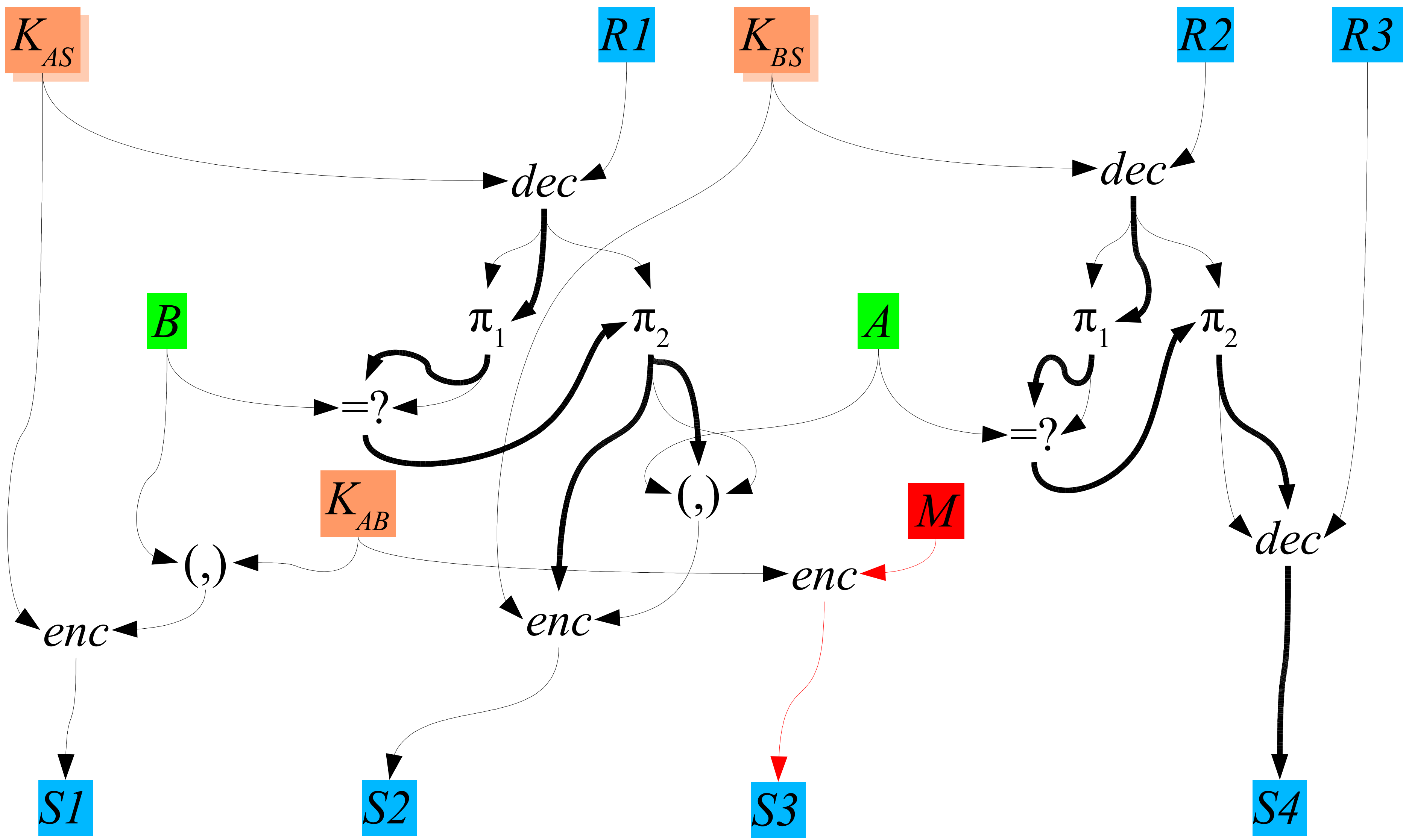
Criterion for security

No path from M to any Si

$$\Downarrow$$

The system is secure

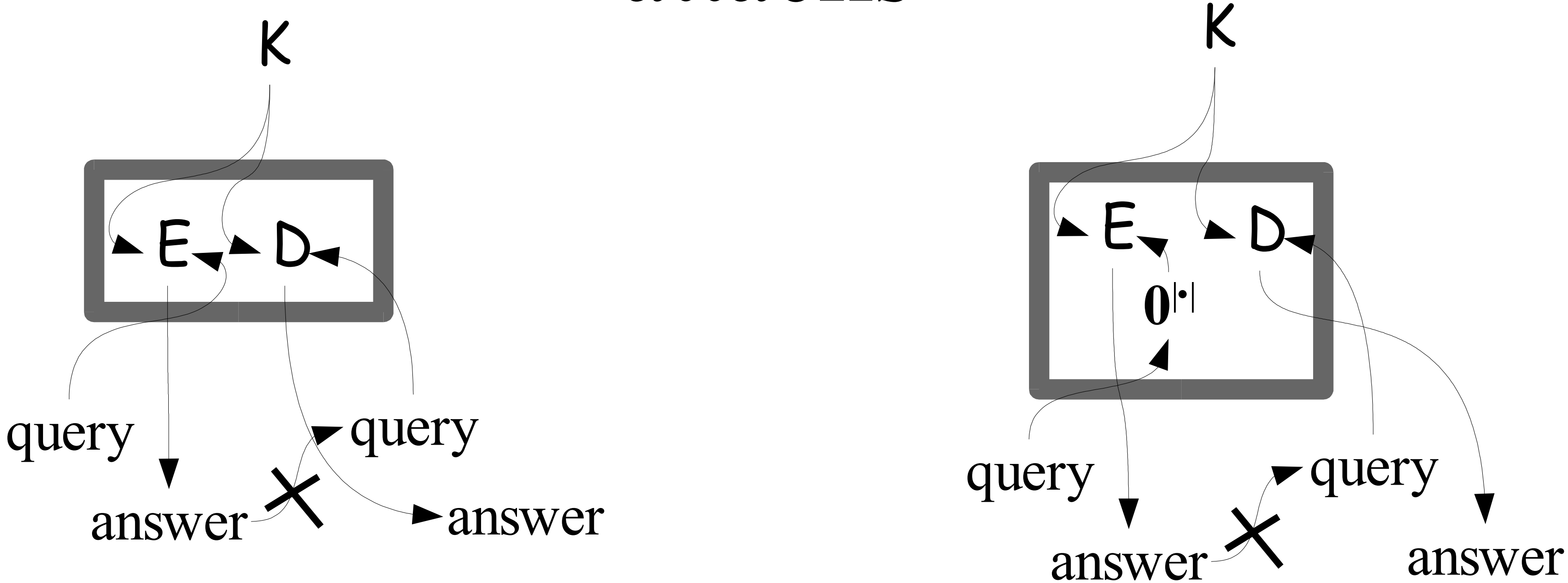
Security does not follow



Encryption systems

- Encryption system is a triple of PPT algorithms:
  - Key generation algorithm K
    - probabilistic
  - Encryption algorithm E
    - may be probabilistic
  - Decryption algorithm D
    - deterministic

Security against chosen-ciphertext attacks



No PPT adversary can distinguish left black box from the right

Without querying the second algorithm with the outputs from the first

In the programming language terms:

We may replace

enc(key: msg)

with

enc(key:const)

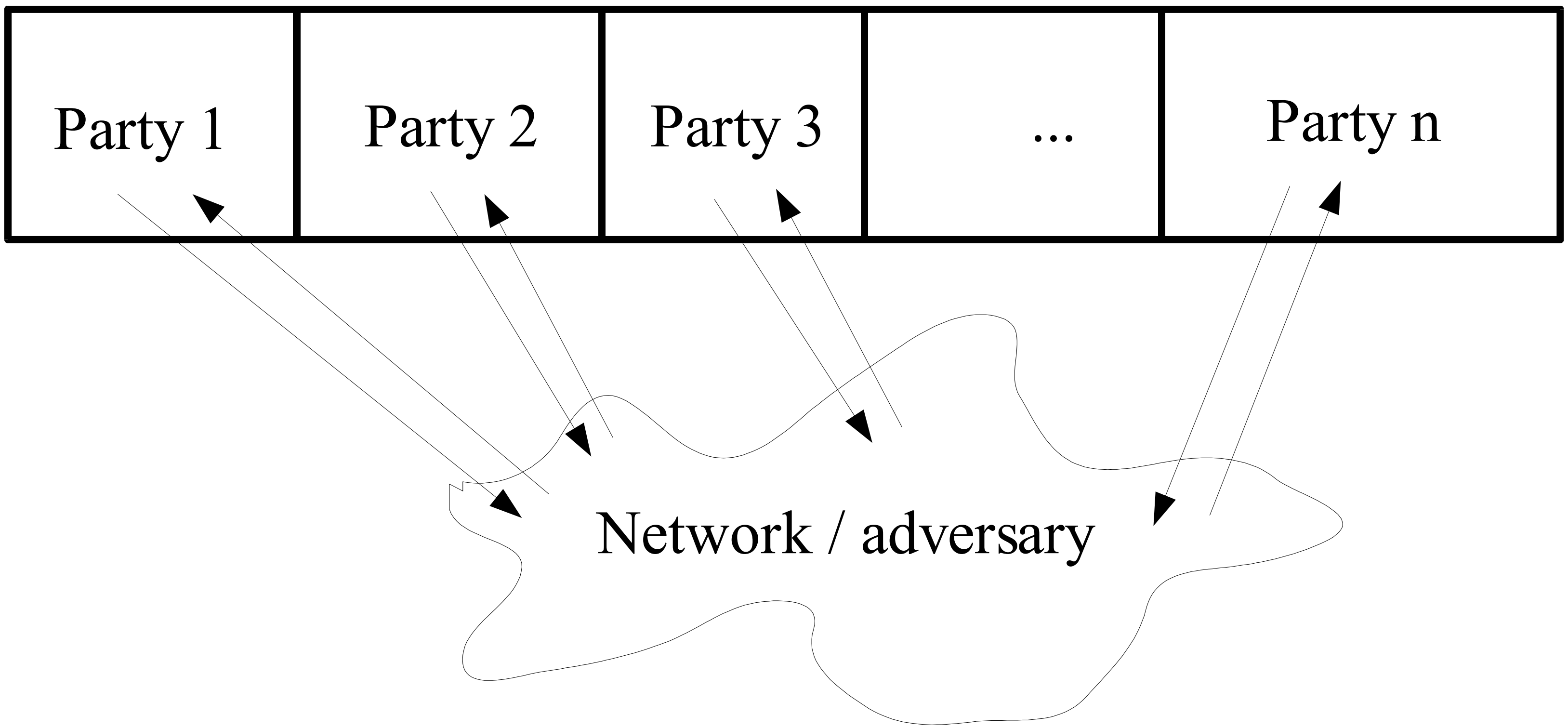
If certain conditions hold then the adversary's view does not change

This replacement deletes a data dependency edge.

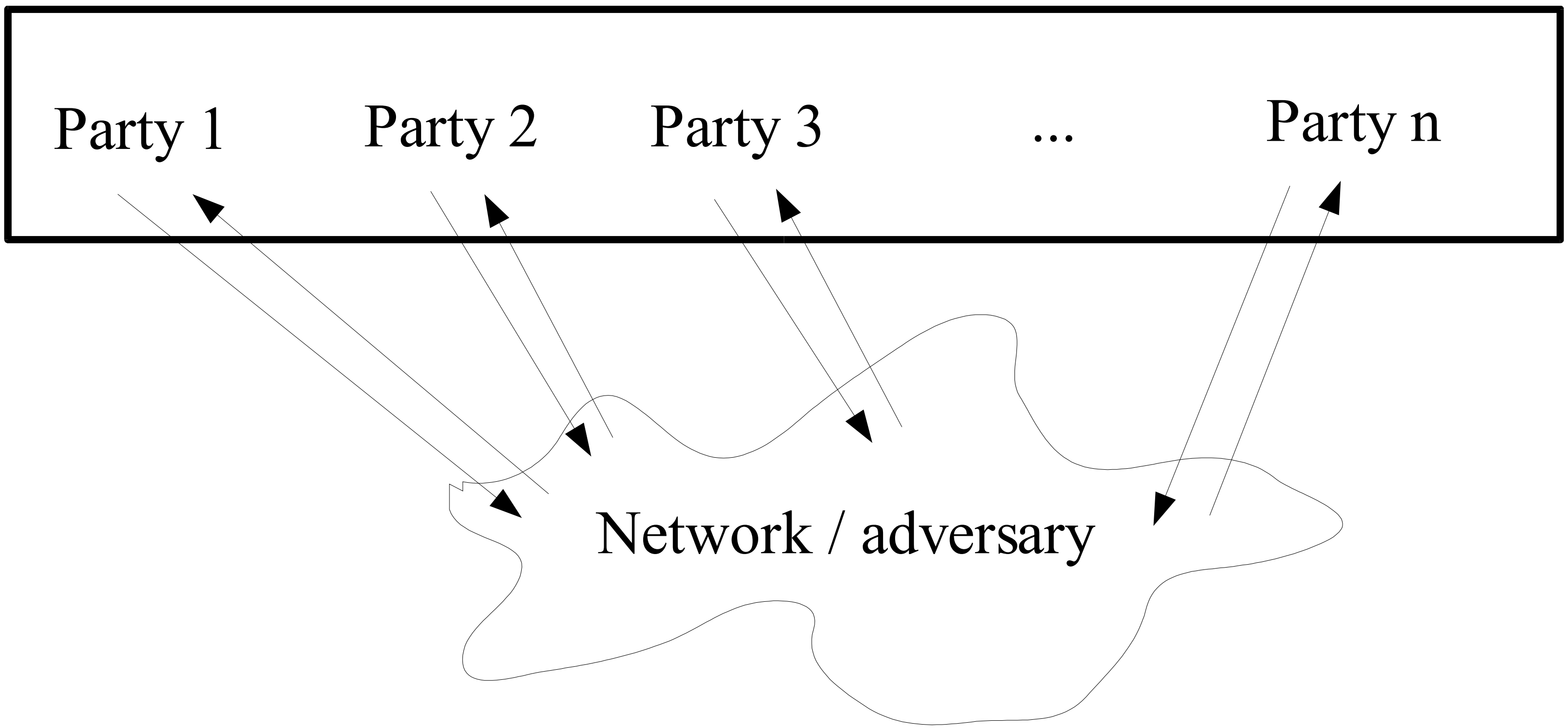
Our contribution

Checking, whether these conditions hold, can be automated.

Use the following intuition...



... all parties are physically together

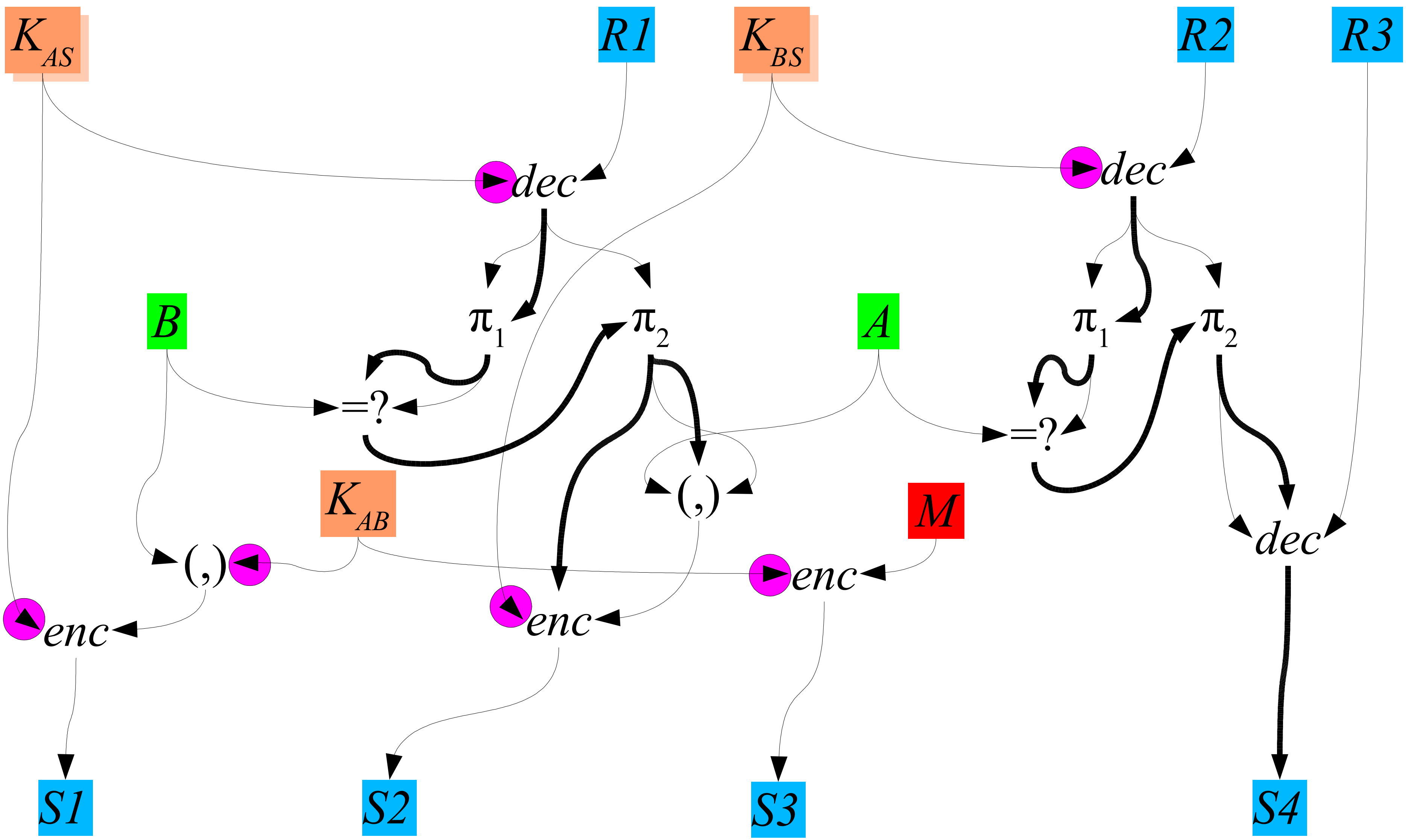


The conditions...

- $enc(K{:}M)$ may be replaced with $enc(K{:}\mathbf{0})$ for <u>all</u> uses of K if

1. K is not really necessary for creating the adversary's view
   - access to oracles $\mathsf{E}_K(\bullet)$ and $\mathsf{D}_K(\bullet)$ must suffice
2. ciphertexts encrypted with K are not subsequently decrypted with it

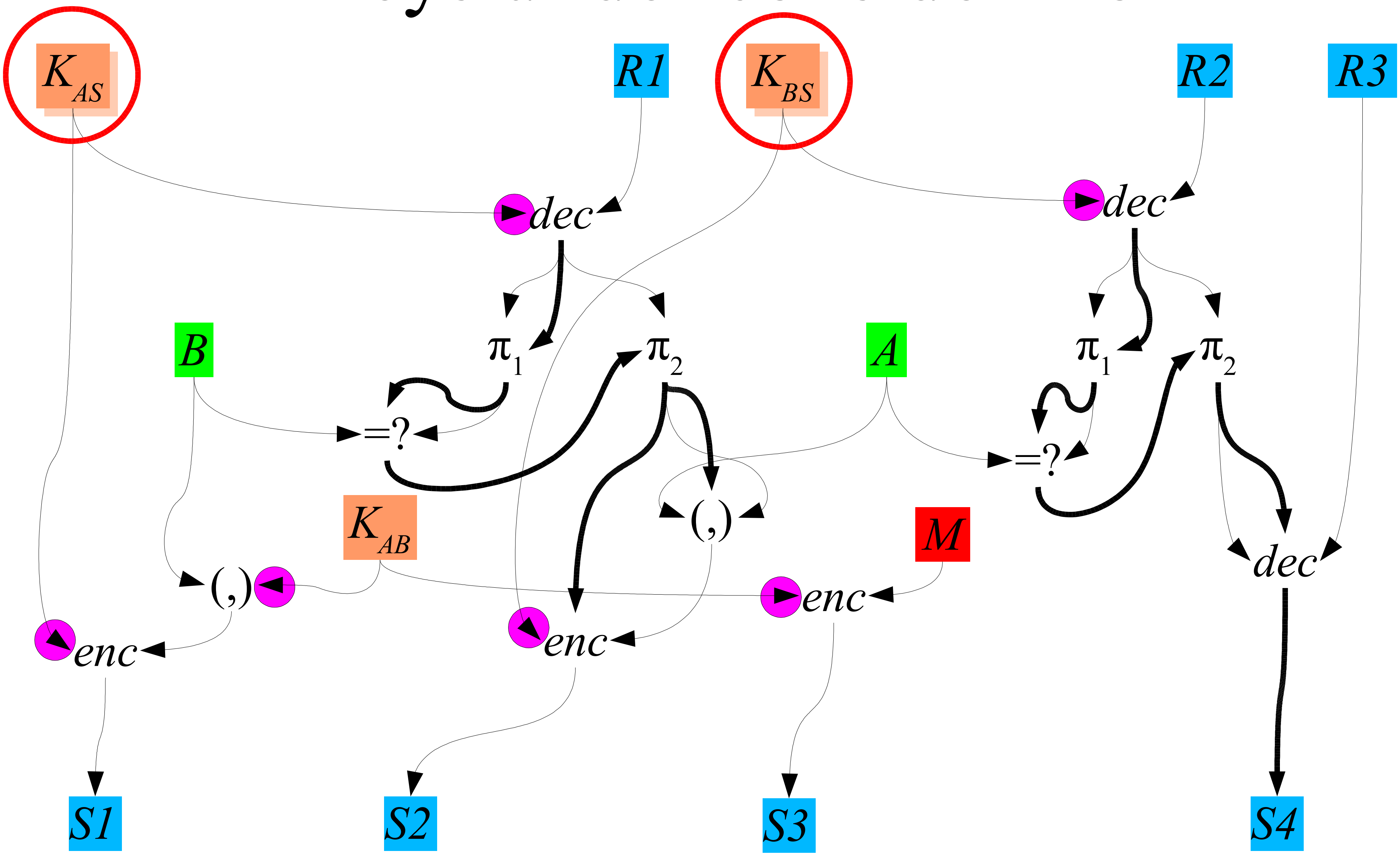
1: find, where the keys are used



1: find, where the keys are used

- Track the values of keys from their generations to their uses
  - Including their flow into and out of constructed values
- Don't consider keys coming from received messages
  - They're ineligible anyway
- Consider only keys used only for encryption and decryption

Keys under consideration



2: replace decryptions

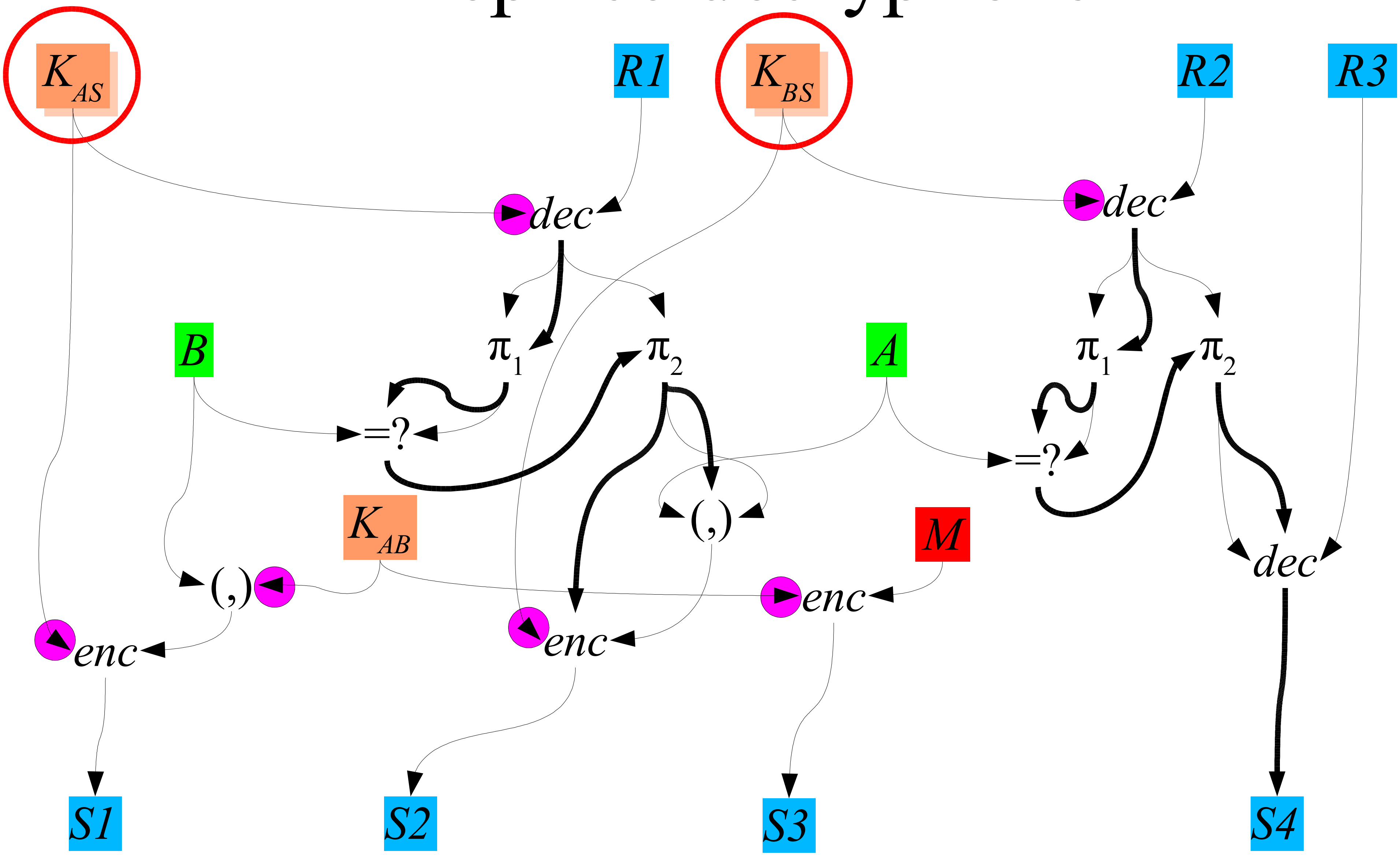
- Let K be a key found in step 1
- Let $y_1,\ldots,y_m$ be the ciphertexts created with K from $x_1,\ldots,x_m$
- Replace $z{:=}dec(K, w)$ with

$$
\begin{array}{lll}
z{:=}\text{case } w \text{ of} & & \\
& \mathrm{y}_1 & \rightarrow x_1 \\
& \ldots\ldots\ldots\ldots\ldots\ldots & \\
& y_m & \rightarrow x_m \\
& else & \rightarrow dec(K, w)
\end{array}
$$

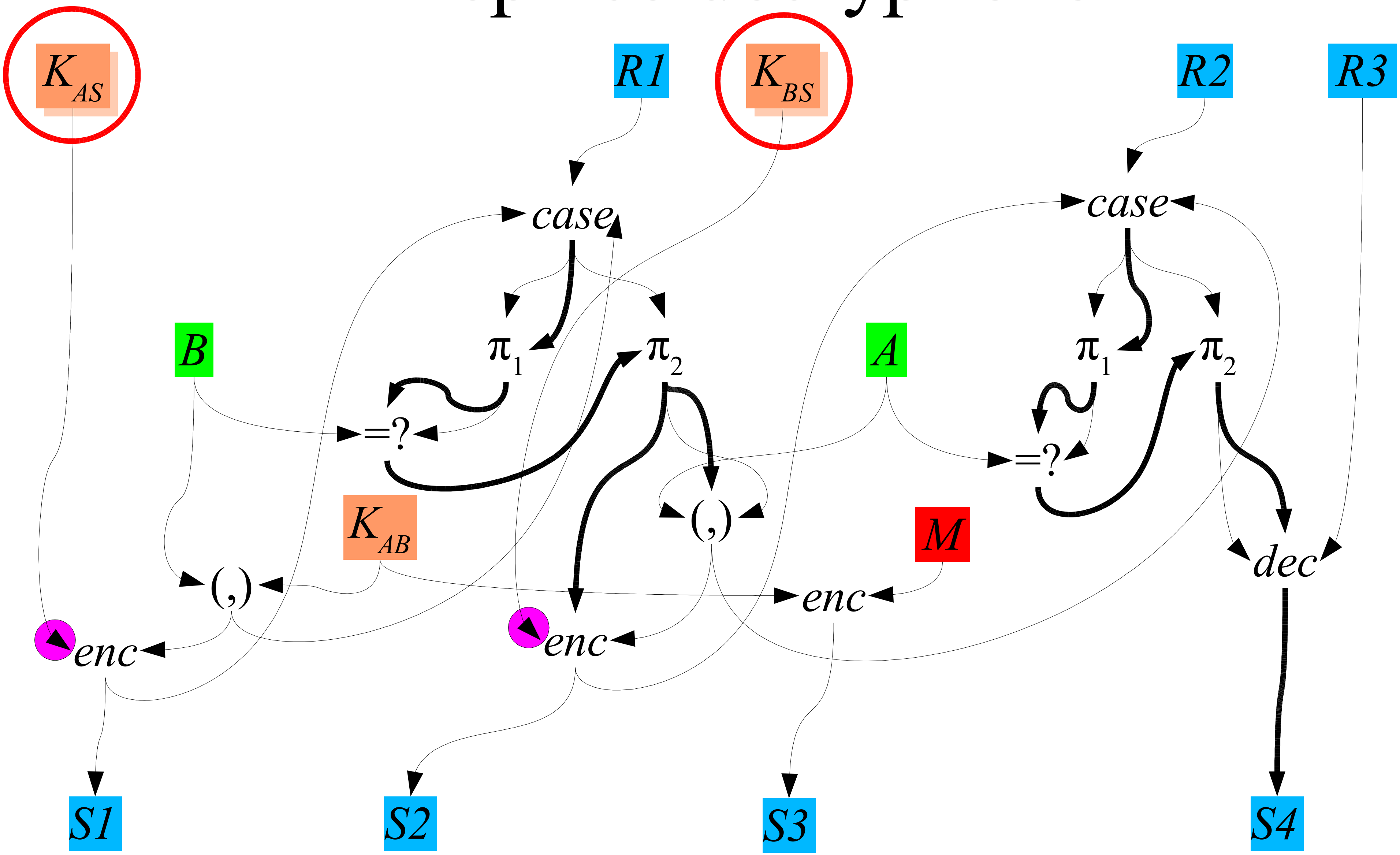
Ciphertext integrity

- No adversary with access to $E_K(\bullet)$ and $D_K(\bullet)$ can create a valid ciphertext different from the ones returned by E
  - Validity: D does not reject it.
- In programming language terms:
  - Remove the *else*-clause in the *case*-statement.

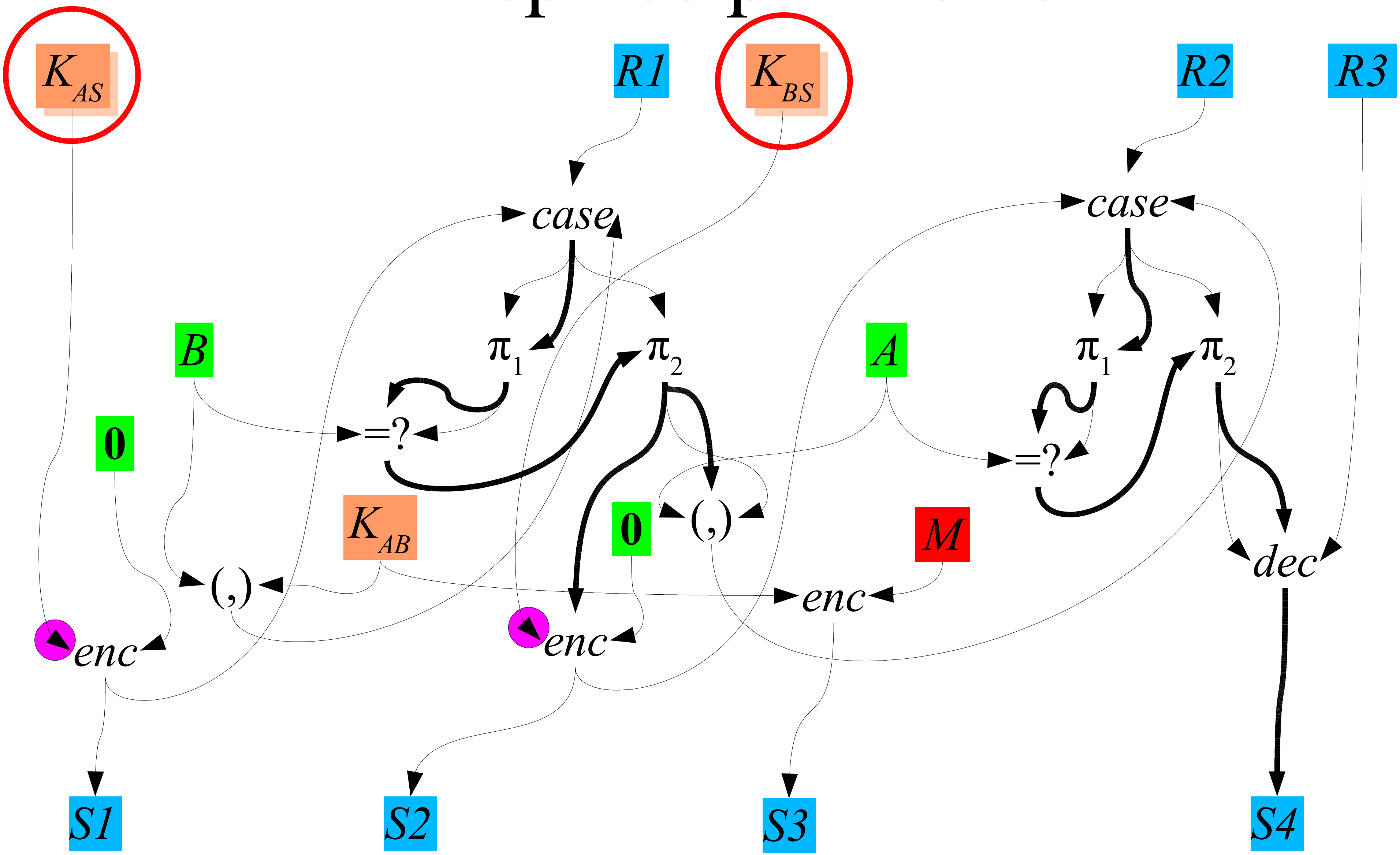
Replace decryptions



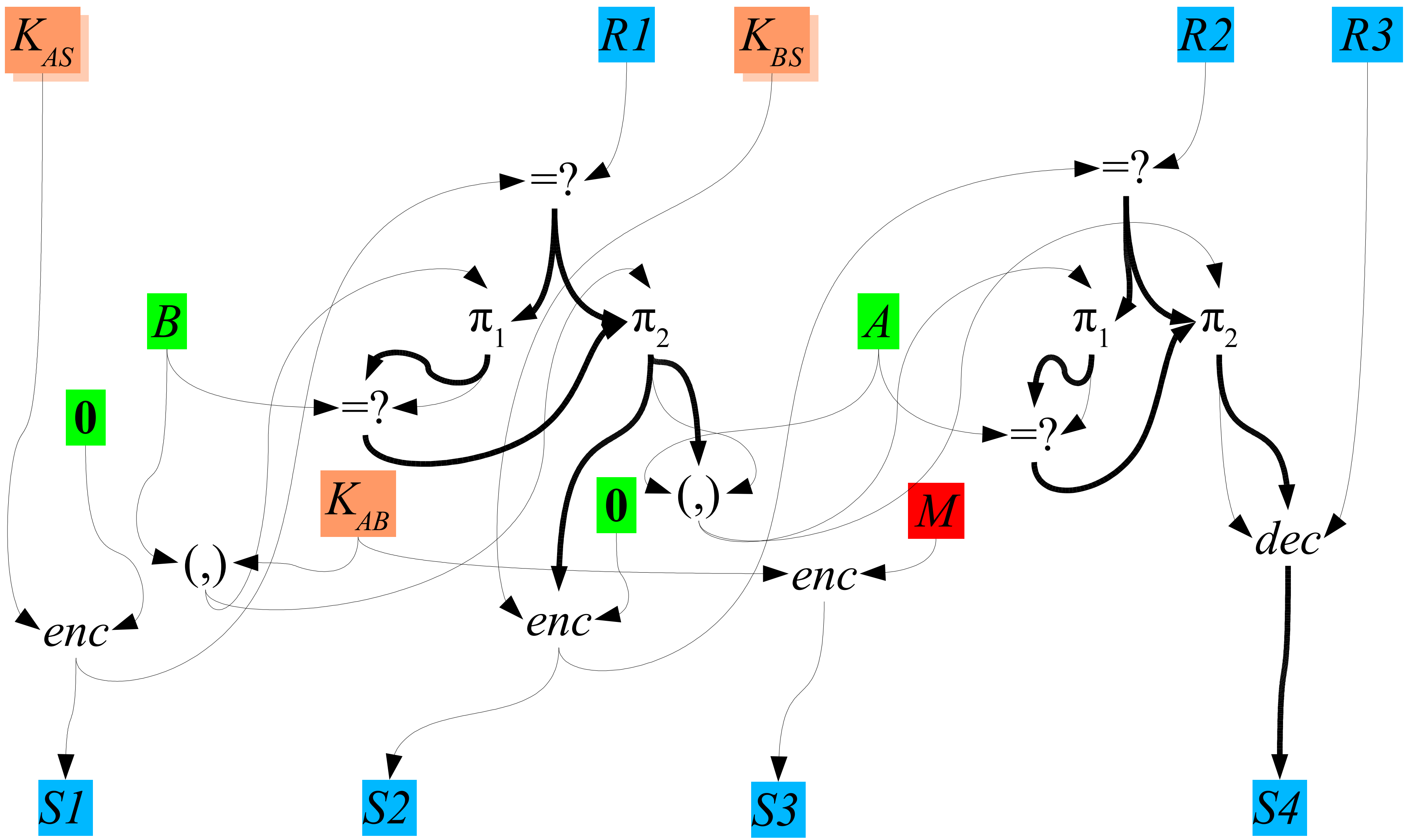
Replace decryptions



Replace plaintexts



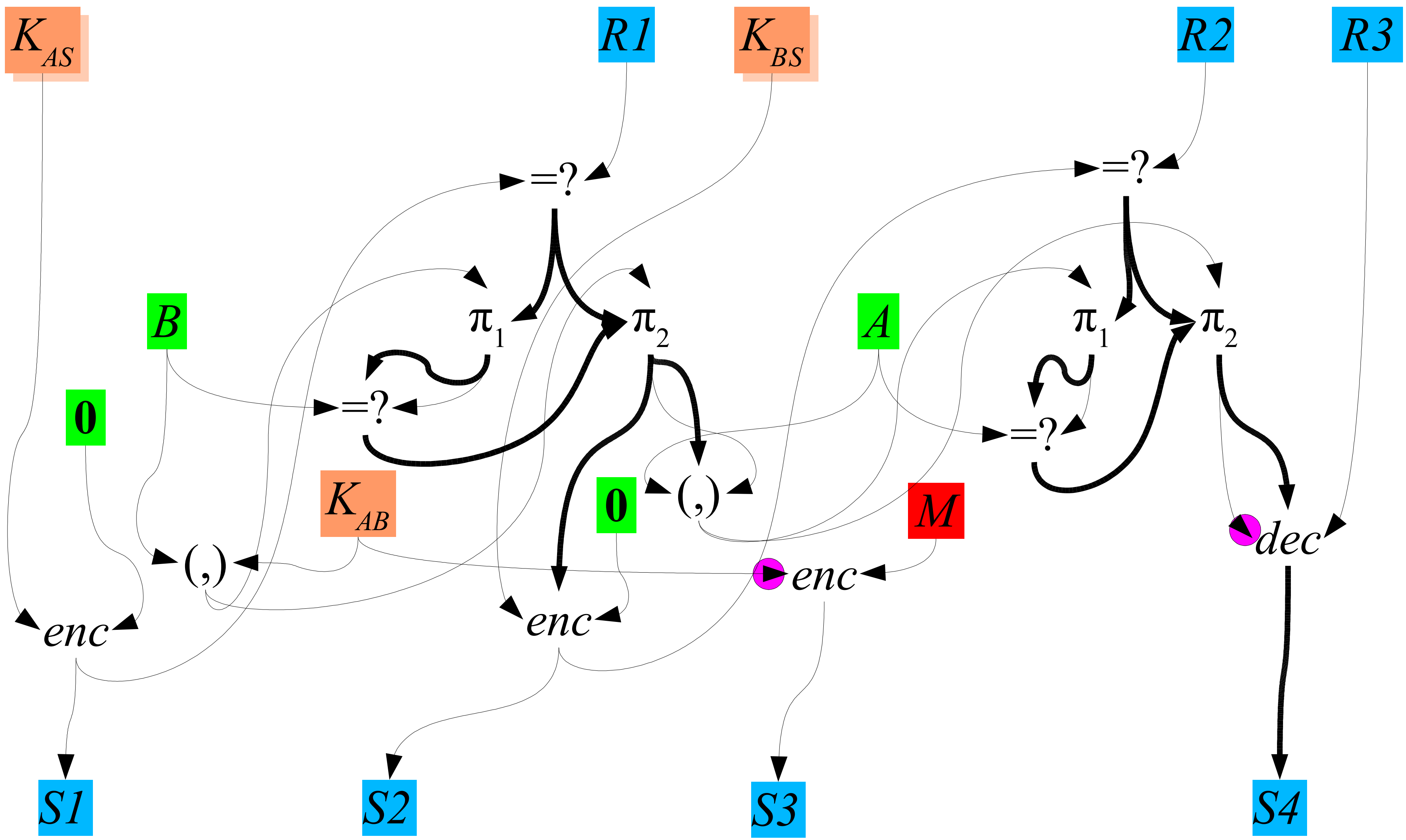
A way to handle *case*-s



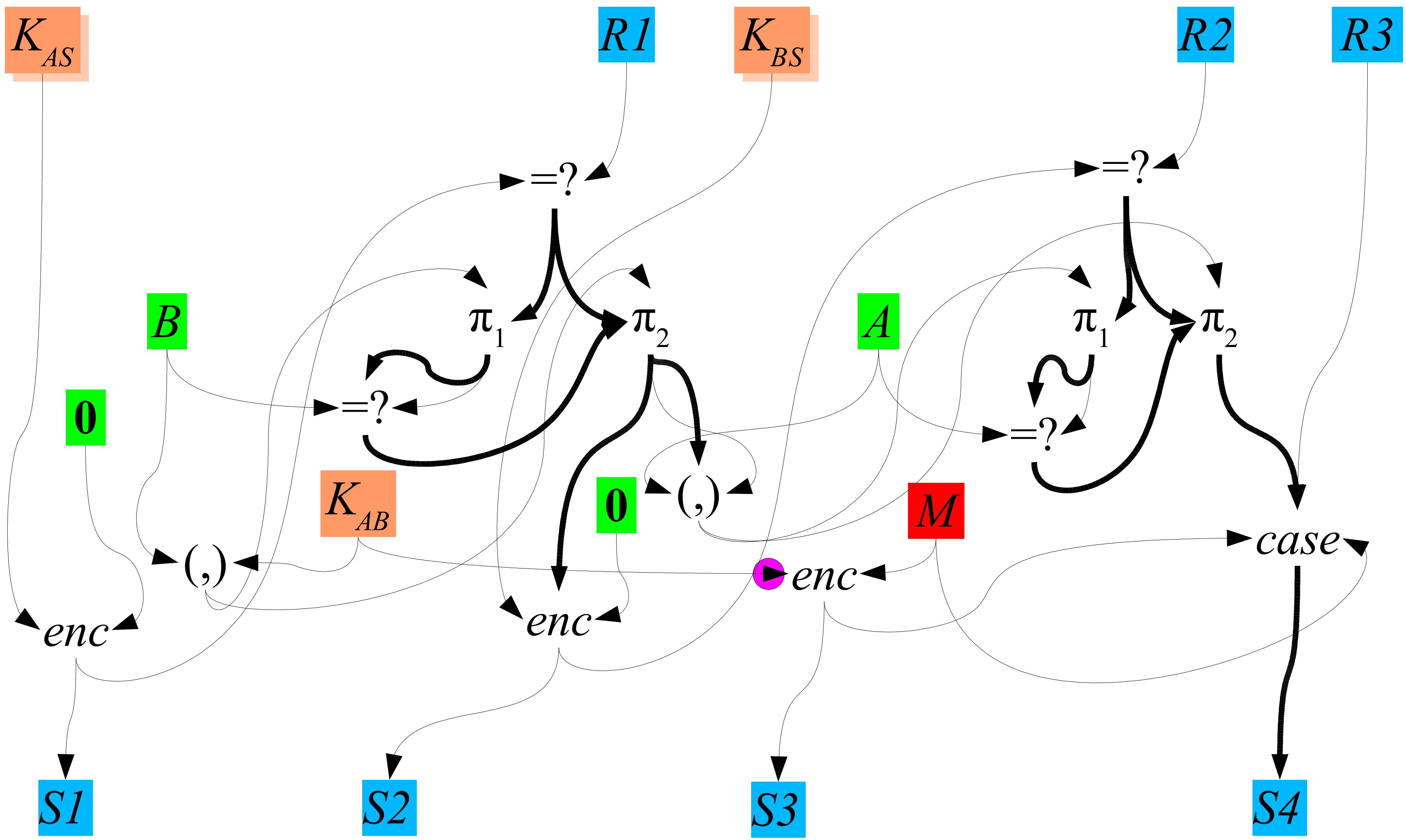
Iterate

- Security does not follow
  - *S3* still depends on *M*
- We try once more
  - In general, do the preceding replacement as long as there are changes.
  - In later iterations do not consider keys that were already handled in previous iterations.

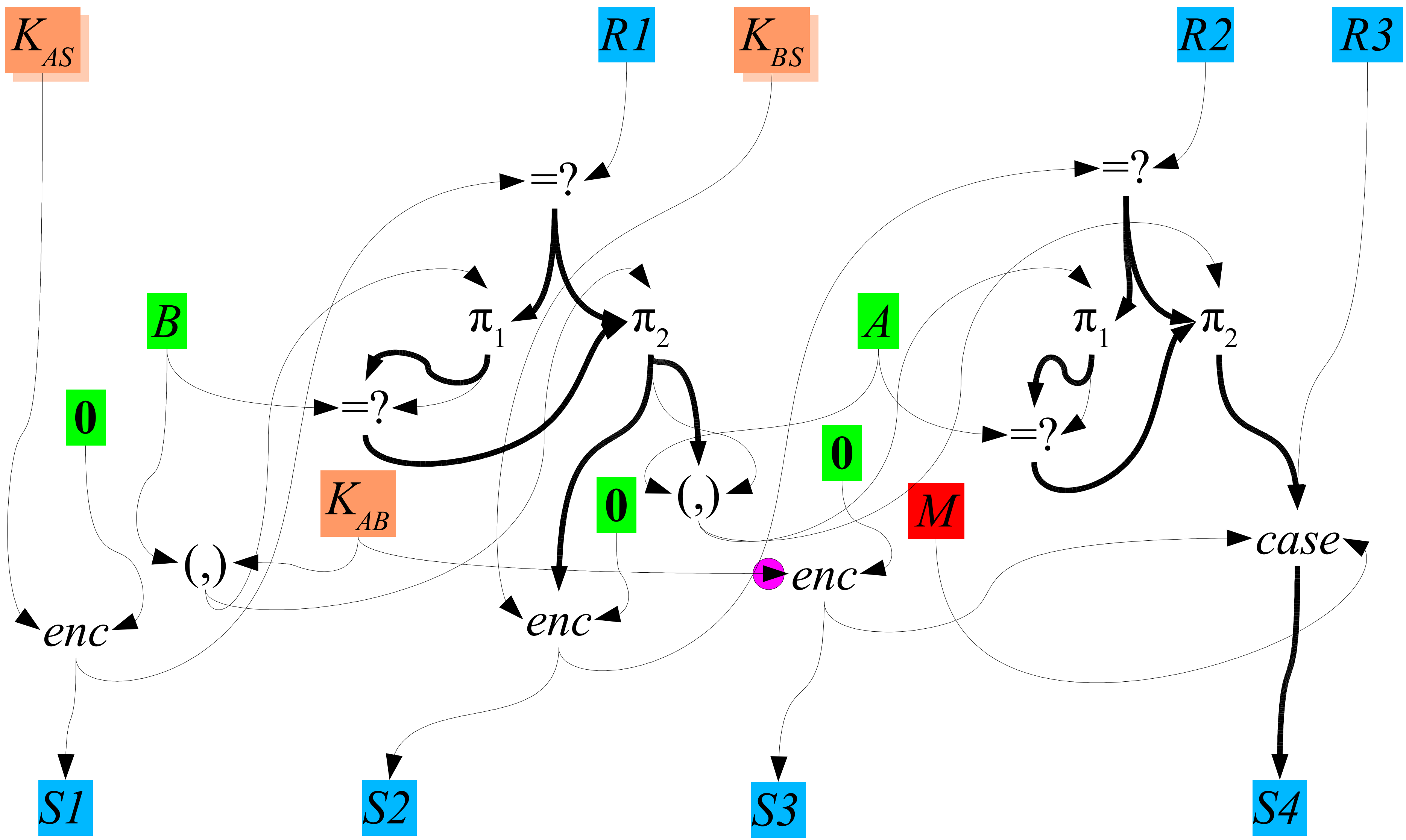
Find, where the keys are used



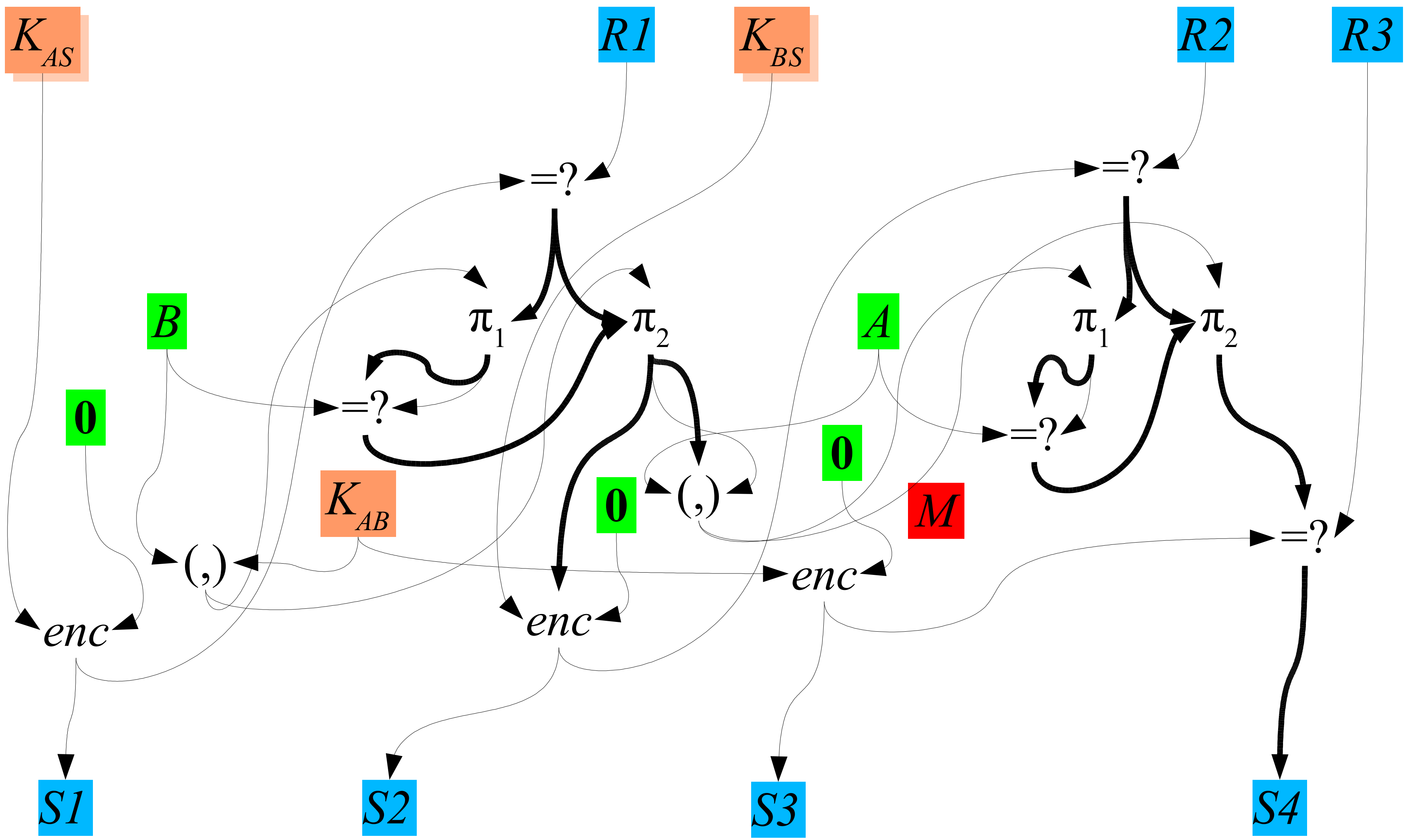
Replace decryptions



Replace plaintexts



Replace *case*, security follows



Generalizability

- Other cryptographic primitives
  - Security def: Indistinguishability of real and ideal functionality
  - Ideal functionality implementable in prog. language
    - Public-key encryption
    - Signatures
    - etc.
- Other security properties
  - Original protocol has the property iff the modified protocol has the property
    - If the adversary can observe violations of the property