**BAN Logic**

**A Logic of Authentication**

(Mike Burrows, Marin Abadi, Roger Needham)

Published 1989, SRC Research Report 39

Presentation by
Heather Goldsby
Michelle Pirtle

---

**Overview**

- Problem
- Solution – BAN Logic
- Goals of BAN
- Terms
- Symbols, Notation, and Syntax
- Example of BAN - Needham Schroder Protocol
- Impact and Limitations of BAN
- Tool Support
- Key Sources

---

**What was the problem?**

- Increased usage of computer networks
- A lack of trust during correspondence
  - Need to know actual sender of messages
  - Need to protect accuracy of sent messages
  - Prohibit interception of messages

---

By Heather Goldsby & Michelle Pirtle
Solution

- **BAN Logic**
  - A formalization of the description and analysis of authentication protocols.

- **Objective of BAN Logic?**
  - It is a logic of authentication
    - Describe knowledge & beliefs of involved parties in authentication in a formal manner
    - Formally analyze the changing knowledge and the beliefs of the parties at each step in the protocol.
  - Logic of authentication allows final protocol states to be made available
  - To provide TRUST among communicating parties

Goals of BAN

- State what is accomplished by the protocol
- Allow reasoning about, and comparisons of, protocol assumptions
- Draw attention to unnecessary actions that can be removed from a protocol
- Highlight any encrypted messages that could be sent in clear text

Terms

- **Idealization:** Used to show central information about beliefs of the recipient in a protocol step.
  - E.g. Clear text parts are omitted in BAN
- **Nonces:** unique number generated for the purpose of being fresh
  - E.g. Timestamp, sequence number
- **Fresh:** never been sent in a message before the current run of the protocol.
- **Time:**
  - past - before protocol began
  - present - any time after protocol began
## Authentication Protocol Syntax

*Message* = *(source, destination, content)*

- **Source**: People / computers / services sending messages
- **E.g.**: Computer A
- **Destination**: People / computers / services receiving messages
- **E.g.**: Server S
- **Content**: The information being sent between a source and a destination
- **E.g.**: Message M: M = “Hello World”
- **Belief**: Things that can be believed by the source and destination but not transmitted
- **E.g.**: Computer A believes that Server S just sent Message M.

## BAN Logic Transformation Process

1. Transform message into idealized logical formula
   - Skip the message parts that do not contribute to the receiver’s beliefs
2. State assumptions about original message
3. Make annotated idealized protocols for each protocol statement with assertions
4. Apply logical rules to assumptions and assertions
5. Deduce beliefs held at the end of protocol

## Symbols

- **Principals**: P, Q, R
- **Specific Principals**: A, B, S
- **Encryption Key**: K
- **Specific Shared Keys**: K_{ab}, K_{aw}, K_{sw}
- **Specific Public Keys**: K_{a}, K_{b}, K_{s}
- **Specific Secret Keys**: K_{a}^{-1}, K_{b}^{-1}, K_{s}^{-1}
- **Statements/Formulas**: X, Y
- **Specific Statements**
  - Nonces: N_a, N_b, N_c
Notation

- $P \models X$: $P$ believes $X$
- $P \bowtie X$: $P$ sees $X$
- $P \models X$: $P$ once said $X$
- $P \models X$: $P$ has jurisdiction over $X$
- $\#(X)$: The formula $X$ is fresh
- $P \bowtie K \cdot Q$: $P$ and $Q$ may use the shared key $K$ to communicate

Notation (cont.)

- $K \rightarrow P$: $P$ has $K$ as a public key
- $P \leftarrow X \rightarrow Q$: Formula $X$ is a secret known only to $P$ and $Q$, and possibly to principles trusted by $P$ and $Q$
- $\{X\}_K$: The formula $X$ is encrypted under the key $K$

Example: Needham Schroder Protocol with shared keys

- What does the Needham Schroder Protocol do?
  - Distributes a secret session key between two principals in a network.
- How the Needham Schroder Protocol works during a threat
  - The protocol assumes the secret key shared with the server is intercepted by the intruder and the intruder can read/make anything passed on the network.
  - The protocol also assumes intruders have the ability to block messages from reaching their destinations and insert malicious messages.
Message 1: \( A \rightarrow S: A, B, N_A \)
- A makes contact with server S stating A wants a key to talk with B, \( N_A \) is fresh.

Message 2: \( S \rightarrow A: \{N_A, B, K_{AB}, \{K_{AB}, A\}_K_S\} \) \( K_S \)
- A message from Server S to principle A consisting of a nonce, key \( K_S \), a statement about the freshness of \( K_S \), and an encrypted version of \( K_{AB} \) to be sent to principle B.

Message 3: \( A \rightarrow B: \{K_{AB}, A\}_K_S \)
- A message from Principle A to Principle B informing B of the key \( K_{AB} \) encoded with the shared key of Principle B and Server S.

Message 4: \( B \rightarrow A: \{N_B\}_K_S \)
- A message from principle B to principle A containing a nonce and \( K_S \), A & B’s shared key.

Message 5: \( A \rightarrow B: \{N_B^{-1}\}_K_S \)
- A message from principle A to principle B consisting of a nonce and \( K_S \).

Example (cont.)

Step 1: Transform message into formal logical formula (Message 1 is skipped.)
- Message 2: \( S \rightarrow A: \{N_A, (A \ast K_{AB} \ast B), \#(A \ast K_{AB} \ast B), \{A \ast K_{AB} \ast B\}_K_S\} \) \( K_S \)
  - Message from S to A encrypted with key \( K_S \)
  - \( N_A \) – A’s nonce indicating freshness of message
  - \( (A \ast K_{AB} \ast B) \) – key \( K_{AB} \) shared between A and B
  - \( \#(A \ast K_{AB} \ast B) \) – nonce indicating key \( K_{AB} \) is fresh
  - \( \{A \ast K_{AB} \ast B\}_K_S \) – key \( K_S \) encrypted with key \( K_S \)

- Message 3: \( A \rightarrow B: \{\ast K_{AB} \ast B\}_K_S \)
  - Message from A to B encrypted with key \( K_S \)
  - \( \ast K_{AB} \ast B \) – key \( K_{AB} \) shared between A and B
Example (cont.)

Step 1: Transform message into idealized logical formula

- **Message 4:** B→A: (N_B, (A • K_{ab} • B))_{K_{ab}}
  - Message from B to A encrypted with key K_{ab}
  - N_B - B's nonce indicating freshness
  - (A • K_{ab} • B) - key K_{ab} shared between A and B

- **Message 5:** A→B: (N_A, (A • K_{ab} • B))_{K_{ab}}
  - Message from A to B encrypted with key K_{ab}
  - N_A - A's nonce indicating freshness
  - (A • K_{ab} • B) - key K_{ab} shared between A and B

Example (cont.)

Step 2: State assumptions about original message

These are all beliefs within the protocol:

- A ⊨ A • K_{ab} • S
  - A believes K_{ab} is a shared key between A and S
- B ⊨ B • K_{ab} • S
  - B believes K_{ab} is a shared key between B and S
- S ⊨ A • K_{ab} • S
  - S believes K_{ab} is a shared key between A and S
- B ⊨ S • K_{ab} • B
  - S believes K_{ab} is a shared key between S and B
- S ⊨ A • K_{ab} • B
  - S believes K_{ab} is a shared key between A and B
- A ⊨ (S ⊨ A • K_{ab} • B)
  - A believes S has jurisdiction over the shared key between A and B

Example (cont.)

Step 2: State assumptions about original message

- B ⊨ (S ⊨ A • K_{ab} • B)
  - B believes S has jurisdiction over the shared key between A and B
- A ⊨ (S ⊨ (A • K_{ab} • B))
  - A believes S has jurisdiction over the freshness of the shared key between A and B
- A ⊨ #(N_A)
  - A believes statement N_A is fresh
- S ⊨ #(A • K_{ab} • B)
  - S believes key K_{ab} is fresh
- B ⊨ #(#(N_B))
  - B believes statement N_B is fresh
- B ⊨ #(#(A • K_{ab} • B))
  - B believes key K_{ab} is fresh
Example (cont.)
Step 3: Make annotated idealized protocols for each protocol statement with assertions

- **Message 2:** $S \rightarrow A: (N_u, (A \ast K_{ab} \ast B), \#(A \ast K_{ab} \ast B), (A \ast K_{ab} \ast B)K_{ab})$ $K_{ab}$

- **Message 2 with Annotations:**
  - $A \ast (N_u, (A \ast K_{ab} \ast B), \#(A \ast K_{ab} \ast B), (A \ast K_{ab} \ast B)K_{ab})$ $K_{ab}$
  - $A$ sees a message encrypted with key $K_{ab}$
  - $N_u \ast A$'s nonce indicating freshness of message
  - $(A \ast K_{ab} \ast B) \ast \#$ key $K_{ab}$ shared between $A$ and $B$
  - $(A \ast K_{ab} \ast B) \ast \#$ nonce indicating key $K_{ab}$ is fresh
  - $(A \ast K_{ab} \ast B)K_{ab} \ast \#$ key $K_{ab}$ encrypted with key $K_{ab}$

Example (cont.)
Step 4: Apply logical rules to assumptions and assertions

- **Nonce-Verification rule:**
  - Checks that a message is recent, thus the sender still believes the message.
  - $P \models \#(X), P \models Q \models X$
  - $P \models Q \models X$
  - If $P$ believes message $X$ is fresh and $P$ believes $Q$ once said $X$ then $P$ believes $Q$ believes $X$

- **Implement Nonce Verification Rule on annotated message 2:**
  - $A \ast (N_u, (A \ast K_{ab} \ast B), \#(A \ast K_{ab} \ast B), (A \ast K_{ab} \ast B)K_{ab})$ $K_{ab}$
  - **Infer:**
    - $\mathbf{A} \models \#(A \ast K_{ab} \ast B), A \models S \models (A \ast K_{ab} \ast B)$
    - $A \models S \models (A \ast K_{ab} \ast B)$ if A believes message $(A \ast K_{ab} \ast B)$ is fresh and A believes S once said $(A \ast K_{ab} \ast B)$ then A believes S believes $(A \ast K_{ab} \ast B)$

Example (cont.)
Step 4: Apply logical rules to assumptions and assertions

- **Jurisdiction rule:**
  - States if a principal has control over a statement and believes the statement other principals should believe the statement
  - $P \models Q \models X, P \models Q \models X$
  - $P \models X$
  - If $P$ believes $Q$ has jurisdiction over message $X$ and $P$ believes $Q$ believes $X$ then $P$ believes $X$

- **Implement Jurisdiction Rule:**
  - Result of Nonce Verification Rule: $A \models S \models (A \ast K_{ab} \ast B)$
  - **Assumption:** $A \models S \models (A \ast K_{ab} \ast B)$
  - **Infer:**
    - $A \models S \models (A \ast K_{ab} \ast B), A \models S \models (A \ast K_{ab} \ast B)$
    - $A \models (A \ast K_{ab} \ast B)$ if A believes S has jurisdiction over the shared key between A and B, and A believes that S believes $K_{ab}$ is the shared key between A and B, then A believes key $K_{ab}$ is the shared key between A and B.
**Example – Final Beliefs**

Step 5: Deduce beliefs held at the end of protocol

- A \( \models A \, \text{K}_{ab} \, B \)
  - A believes \( \text{K}_{ab} \) is a shared key between A and B
- B \( \models A \, \text{K}_{ab} \, B \)
  - B believes \( \text{K}_{ab} \) is a shared key between A and B
- A \( \models B \, \text{K}_{ab} \, B \)
  - A believes that B believes \( \text{K}_{ab} \) is a shared key between A and B
- B \( \models A \, \text{K}_{ab} \, B \)
  - B believes that A believes \( \text{K}_{ab} \) is a shared key between A and B

**The Impact of BAN**

- First protocol specification language to use formal verification to model authentication
- BAN introduced a simple and powerful notation
- BAN logic postulates (i.e., Nonce-verification rule) are straightforward to apply for deriving BAN beliefs
- BAN logic is the foundation of other protocol specification languages that are more expressive
  - E.g., GNY

**Limitations of BAN**

- Conversion to idealized form
- Lack of ability to state something a principle does not know
  - I.e., private information is not guaranteed to remain private
- Example given by Nessett:
  - Assume principles A and B communicate with public keys
    - A \( \rightarrow B \): [\( \text{K}_{ab} \) A], 1
    - B \( \rightarrow A \): [\( \text{K}_{ab} \) A]
  - In idealized form:
    - A \( \rightarrow B \): [\( \text{K}_{ab} \) A \( \ast \text{K}_{ab} \) + B] 1
      - A sends B a message containing secret key \( \text{K}_{ab} \) encrypted under A’s private key. The public key is well known making the secret key public knowledge.
    - B \( \rightarrow A \): [A \( \ast \text{K}_{ab} \) + B] 1
      - Message from B to A with the believed shared key \( \text{K}_{ab} \)
  - Result: The originally secret key \( \text{K}_{ab} \) is known and the message between B and A can be read and forged
**Limitations**

- BAN does not catch all protocol flaws
  - False-positives can result
- A principal’s beliefs cannot be changed at later stages of the protocol
  - No division of time in protocol run
- Provides a proof of trust on part of principles, but not a proof of security
  - Final beliefs can be believed only if all original assumptions hold true
- BAN does not account for improper encryption

**Tool Support**

- SPEAR
  - Model analyzer for BAN Logic
  - Developed for security protocols
- Aspects of protocol development SPEAR supports:
  - Protocol specification –
    - Stating possessions
    - Define interactions with external functions and generated source code
    - Define BAN logic beliefs
    - Security analysis – detect possible attacks
    - Code generation – generates Java code
    - Meta execution and performance evaluation – testing the generated Java code on a safe platform

---

**SPEAR (cont.)**

- Possessions types
  - Asymmetric key – use for asymmetric encryption using private and public key pair
  - Symmetric key – used for symmetric encryption using the same key to encrypt and decrypt a message
  - Delimited data – inserts delimiters into sent messages
  - Entity information – used to send identifying information
  - Fixed length data – only allows data of a fixed size
  - Variable length data – allows data of unknown size, such as messages
- The possessions are then initialized by the entities
**SPEAR (cont.)**

- Steps for protocol specification
  - Use Protocol-Set Protocol Name option to set the protocol’s name
    - Not needed for BAN but possible with SPEAR
  - Declare possessions used in protocol
  - Generate needed macros for protocol
    - Not needed for BAN but possible with SPEAR
  - State initial BAN beliefs
    - Done only if BAN analysis is desired
  - Define entities (principals) involved in protocol
  - Initialize possessions required by each entity throughout the duration of the protocol
  - Add messages and statement blocks to the protocol for running functions at protocol stages
    (This is modeling the system)

**Limitations of SPEAR**

- SPEAR is not perfect
  - Shanthoshi and Shreyas found a bug in the belief derivation logic of SPEAR
- Variations between SPEAR and BAN
  - Slightly different syntax
    - BAN: A believes K is fresh
    - SPEAR: A believes fresh(K)
  - Shared symmetric keys in initial beliefs are not always allowed
    - Shanthoshi and Shreyas implemented the shared symmetric keys as public keys to obtain desired results

---

By Heather Goldsby & Michelle Pirtle
### BAN Logic Online Sources

#### The original paper:
- "A Logic of Authentication" by Burrows, Abadi & Needham  
  [http://citeseer.nj.nec.com/burrows90logic.html](http://citeseer.nj.nec.com/burrows90logic.html)

#### Overviews of BAN Logic:
- "A Logic of Authentication by Burrows, Abadi and Needham" by Kynata  
- The BAN Logic of Authentication by Botting  
  [http://www.csci.csusb.edu/dick/samples/BAN.html](http://www.csci.csusb.edu/dick/samples/BAN.html)
- A Semantics for BAN Logic by Bleeker  
  [http://dimacs.rutgers.edu/Workshops/Security/program2/bleek.html](http://dimacs.rutgers.edu/Workshops/Security/program2/bleek.html)
- Lecture on BAN by Wing  

#### Limitations of BAN Logic
- On a Limitation of BAN by C. Boyd & W. Mao  
  [http://citeseer.nj.nec.com/boyd93limitation.html](http://citeseer.nj.nec.com/boyd93limitation.html)
- On a Limitation of BAN by C. Boyd & W. Mao  
  [http://citeseer.nj.nec.com/boyd93limitation.html](http://citeseer.nj.nec.com/boyd93limitation.html)

### BAN Logic Online Sources

#### Tool Support for BAN Logic
- "Automated BAN Analysis of Authentication Protocols" by Santhosh D.B. and Doshi Shreys  
  [http://www.cs.uci.edu/~sdoshi/w01/AutomatedBANAnalysis.pdf](http://www.cs.uci.edu/~sdoshi/w01/AutomatedBANAnalysis.pdf)
- "SPEAR: a Security Protocol Engineering & Analysis Resource" by Santosh D. and Doshi Shreys  
  [http://dimacs.rutgers.edu/Workshops/Security/program2/hutch/spear.html](http://dimacs.rutgers.edu/Workshops/Security/program2/hutch/spear.html)
- Obtain a copy of SPEAR from:  
- "Evaluating Cryptographic Protocols" by A. Yasinsac & W. Wulf  
  [http://citeseer.nj.nec.com/yasinsac93evaluating.html](http://citeseer.nj.nec.com/yasinsac93evaluating.html)

#### Comparison of Cryptographic Protocol Analyses
- "Three Systems for Cryptographic Protocol Analysis" by Kemmerer, Meadows, & Miller  

#### Industrial Use of BAN
- "On BAN logics for Industrial Security Protocols" by Agray, van der Hook, and de Virk  
  [http://www.cs.uu.nl/groups/IS/archive/wiebe/agray.pdf](http://www.cs.uu.nl/groups/IS/archive/wiebe/agray.pdf)