CS 5430

Protocols

Prof. Clarkson
Spring 2016

Review: Secure channel

- When we last left off, we were building a secure channel
  - The channel does not reveal anything about messages except for their timing and size (Confidentiality)
  - If Alice sends a sequence of messages m1, m2, ... then Bob receives a subsequence of that, and furthermore Bob knows which subsequence; and the same for Bob sending to Alice (Integrity)
- Cryptography employed:
  - Authenticated encryption to protect confidentiality and integrity
  - Message numbers to further protect integrity
  - Key establishment protocol to create shared session key
- We built and broke three key transport protocols so far...

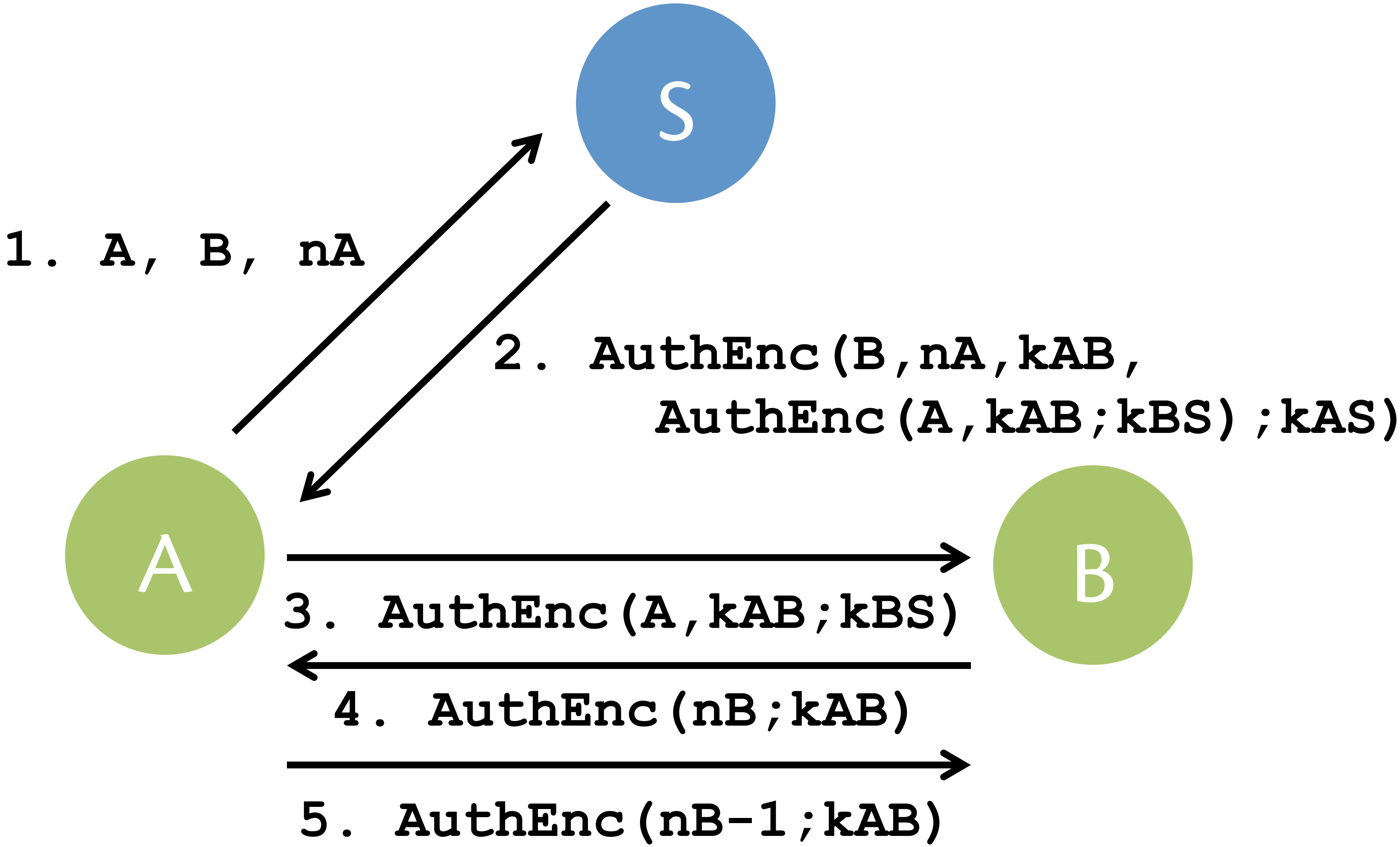
Review: Transport protocol

- **Assume:** trusted server S with whom A and B already share a long-term key
  - A shares kAS with S
  - B shares kBS with S
- **Output:** new session key kAB generated by S
- **Goals:**
  1. Only A, B, and S know that key (confidentiality)
  2. Users associate key with correct principal identities (integrity)
  3. The session key is fresh (integrity)

Review: Challenge-Response

- Challenger issues question
- Responder gives answer
- Example: *From Russia with Love*
  - Unfortunately, that *static challenge* can be replayed
  - So crypto protocols use nonces; parties contribute nonces to be convinced of *freshness*

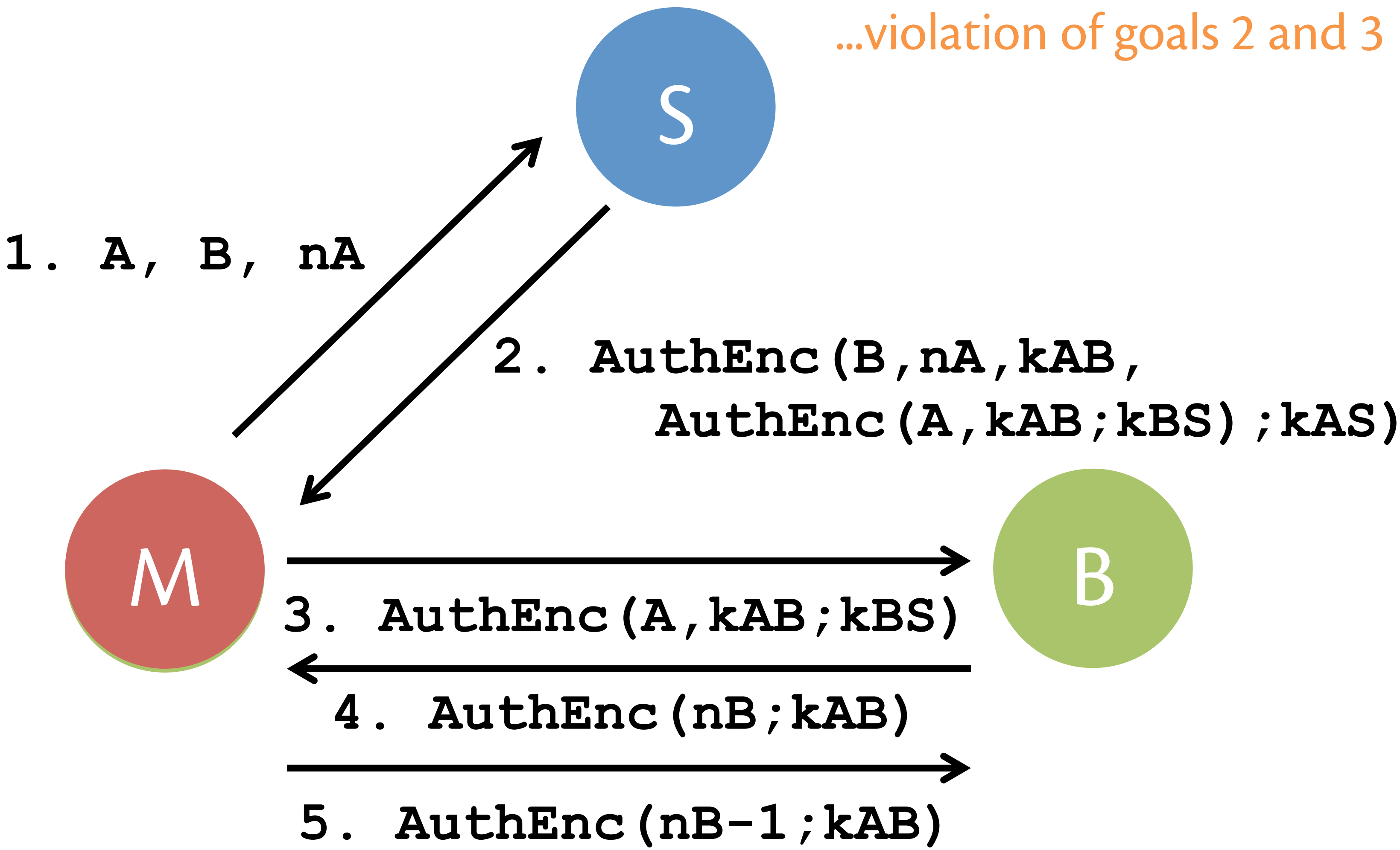
Protocol 4 [Needham & Schroeder 1978]



Protocol 4: Attack 4

Weakness: what if kAB is eventually disclosed to M?

...violation of goals 2 and 3

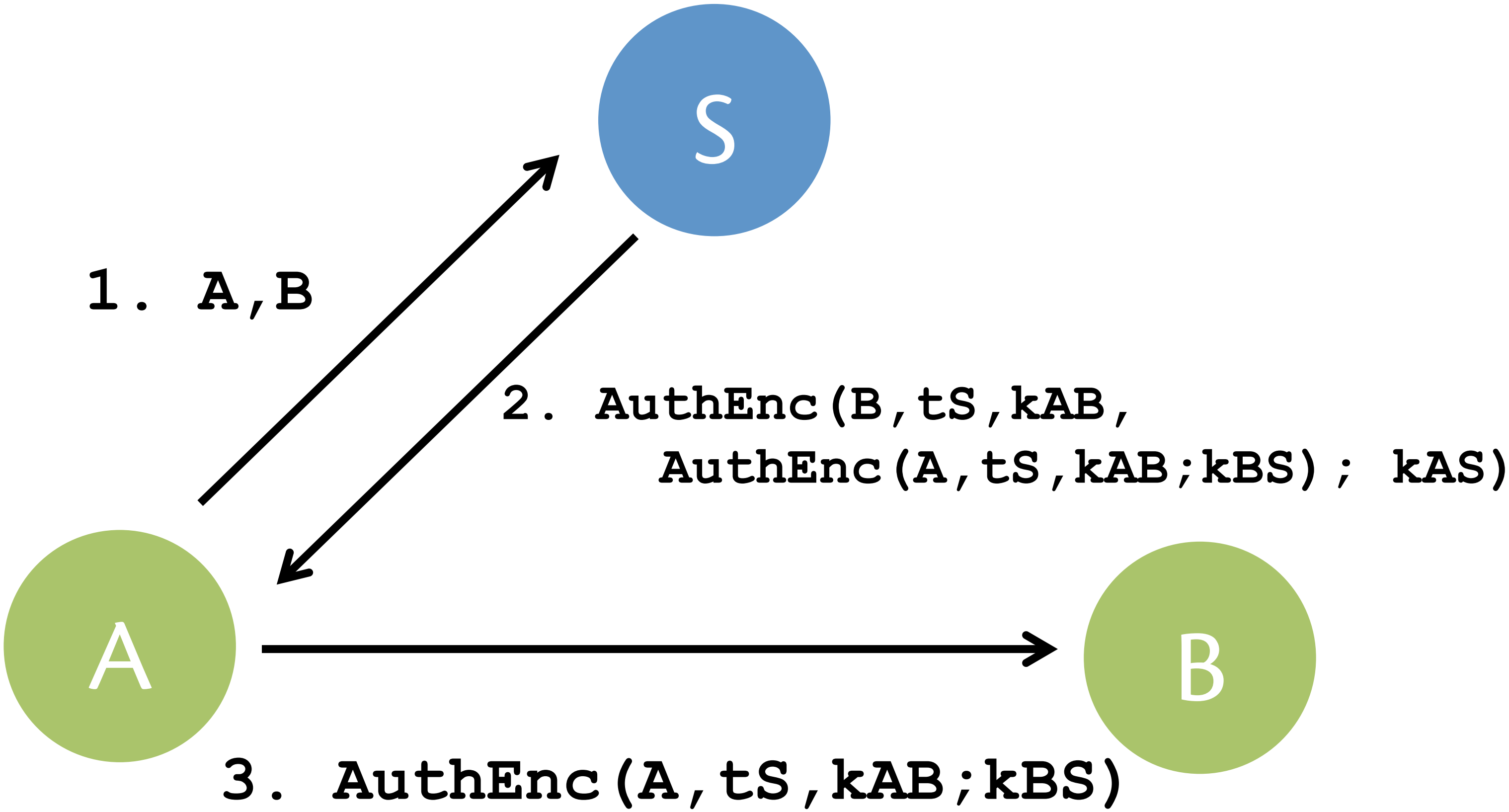


Digression: key erasure

- Is it really likely that adversary learns session key kAB but not any long-term shared keys?
  - Maybe not
  - But we are conservative
- Never assume that deallocation or garbage collector will make keys inaccessible
- **Implementation:** zero out arrays containing keys, passwords, other secrets

Protocol 5 [Denning & Sacco 1981]

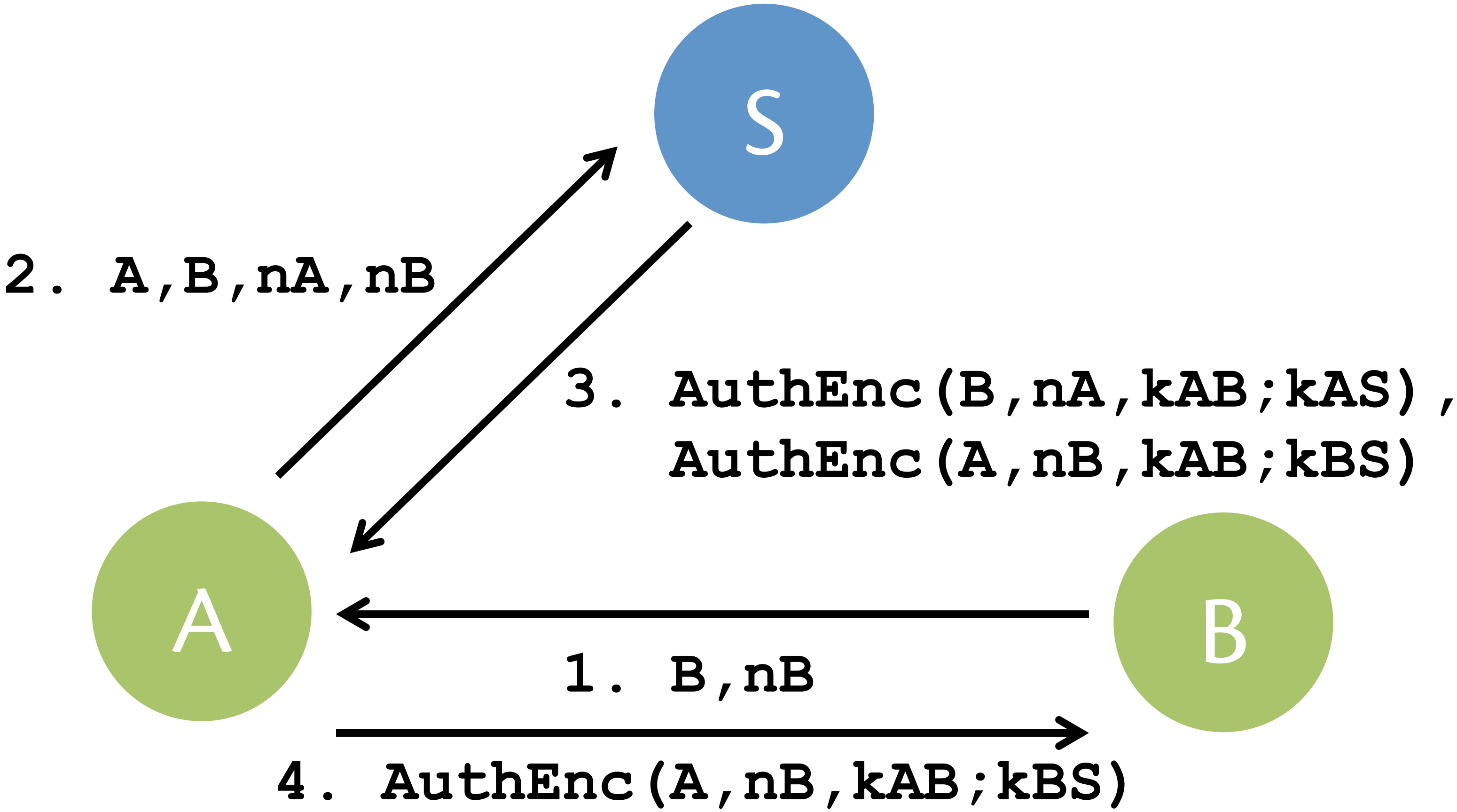
Solution 1: use synchronized clocks and timestamps



tS is time at server S. A and B reject any message that is too old.

Protocol 6 [Bauer et al. 1983]

Solution 2: submit nonces from both users to S

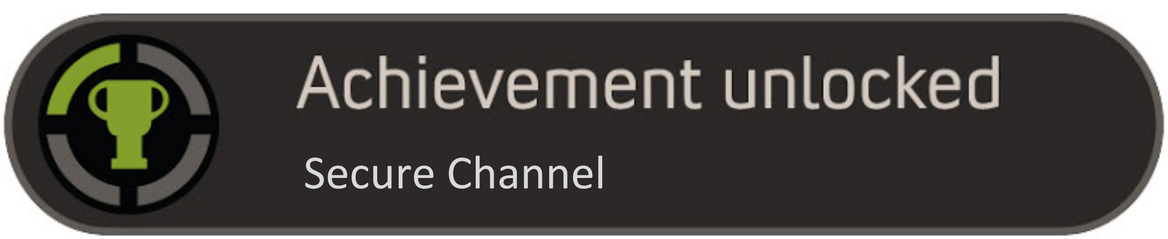


Lessons learned

- Designing simple cryptographic protocol is hard
  - Attacks aren't obvious
  - Published protocols later found to be flawed
- Goals aren't immediately obvious
  - We ended up with three
  - There are many more contemplated in literature

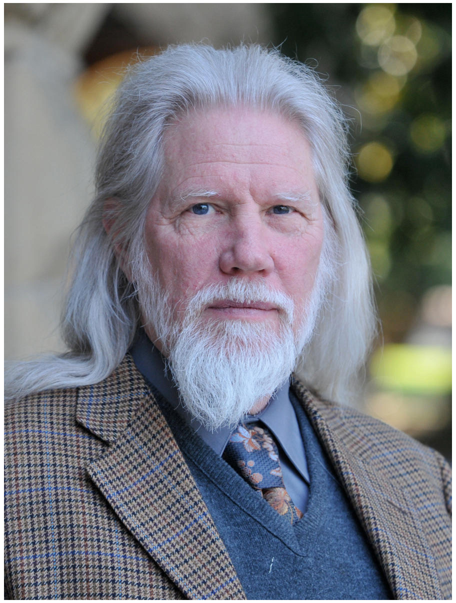
Secure channel

- Use authenticated encryption, message numbers, key establishment protocol
  - assuming users share a long-term key with trusted third-party server
  - other solutions based on asymmetric keys...as in A3
- Now we can have secure conversations!



News flash: Turing Award

Whitfield Diffie and Martin Hellman win the 2015 Turing Award!



For critical contributions to modern cryptography.

The ability for two parties to communicate privately over a secure channel is fundamental for billions of people around the world. On a daily basis, individuals establish secure online connections with banks, e-commerce sites, email servers and the cloud. Diffie and Hellman's groundbreaking 1976 paper, "New Directions in Cryptography," introduced the ideas of public-key cryptography and digital signatures, which are the foundation for most regularly-used security protocols on the Internet today.

Diffie-Hellman(-Ralph Merkle)

- Key agreement protocol [1976]: basis of many later protocols, and still available in SSL, but itself lacks any authentication of principals
- Metaphor based on colors:

https://www.youtube.com/watch?v=YEBfamv-_do&feature=youtu.be&t=138

ATTACKS ON PROTOCOLS

Attacks

- Compare:

  "this protocol resists the following list of attacks"

  vs.

  "this protocol achieves the following security goals under the following assumptions"

- Both establish trustworthiness of protocol
  - latter is more useful to user of protocol
  - former is nonetheless useful to us as analysts

Attacks

- **Eavesdropping:** passively capture messages
  - primary countermeasure: encryption
- **Replay:** record and resend messages
  - maybe to same or different principal
  - maybe in same or different protocol run
  - primary countermeasure: nonces (counters, timestamps, random numbers)
  - special cases:
    - **Preplay:** attacker engages in early protocol runs with goal of attacking later run
    - **Reflection:** send protocol messages back to principal who originally sent them, often in parallel runs with flipped roles
    - **Man in the middle:** attacker interposes between two principals, perhaps pretending to be the other to each of them

Attacks

- **Modification:** actively alter messages
  - many attacks don't alter fields of message but splice together fields from separate messages
  - primary countermeasure: MACs, which must tie together fields to prevent splicing
  - **Typing attack:** cause principal to mis-parse message, e.g. interpret a principal identifier as a key or v.v.
- **Protocol:** attacker can run whatever protocol it wants
  - maybe another protocol that uses the same keys in a different way (which might violate our principle of using keys for unique purposes)
  - maybe a custom-designed protocol

PRINCIPLES FOR PROTOCOLS

Design principles

[Abadi and Needham 1995]

- Wisdom derived from analysis of many protocols and attacks
- Not sufficient to guarantee security
- Not necessary to guarantee security
- But following principles would have prevented mistakes

Say what you mean

Main principle: Every message should say what it means

- Interpretation of message should depend only upon content of message
- Hence recipient can recover meaning without needing to assume or supply any context
- Writing down a straightforward English sentence describing the meaning of each step in narration is good practice

Say what you mean

Protocol narrations sometimes work against this principle:

Example: **`4. B -> A: X`**

- Protocol designer intended...
  - it's the fourth message sent
  - the contents are X
  - B originates it
  - A receives it
- Because of attacker, none of those is necessarily true

Say what you mean

Protocol narrations sometimes work against this principle:

Another example: `S -> A: Enc(B,kAB; kAS)`

- Might mean *"S sends to A a session key kAB intended to be good for conversation with B"*
- But the narration itself doesn't say that clearly
- And if it were `S -> A: Enc(kAB; kAS)`, then A would have to guess that the key is for B, or assume it from context of other messages in protocol

Say what you mean

Two forms of confusion:

- between current message expected by principal, and same message from previous run of same protocol
- between current message and different message in protocol or from different protocol

Say what you mean

Principle: Message contents should describe what protocol, which instance, and message number in it

Example (back to Protocol 4), instead of:

```
4. B->A: AuthEnc(nB;kAB)
5. A->B: AuthEnc(nB-1;kAB)
```

could verbosely use:

```
4. B->A: AuthEnc("NS4",A,B,kAB,nB;kAB)
5. A->B: AuthEnc("NS5",A,B,kAB,nB;kAB)
```

Naming

Principle: Explicitly name the relevant principals in each message

- If principals are not named, recipient has to make assumptions from context
- Assumptions are vulnerabilities
- Attacker will exploit with replay, modification attacks

Naming

Example [Denning and Sacco 1981]:

```
1. A -> B: Enc(kAB,tA,Sign(kAB,tA;k_A);K_B)
```

Intended meaning might be *"At time tA, principal A says that kAB is a good key for communication between A and B"*

- But the message doesn't name A or B
- Maybe it's okay not to name A, since A's private key is used
- But there's an attack that's possible because B is not named...

Naming

M gets A to start a protocol run...

```
1. A -> M: Enc(kAM,tA,Sign(kAM,tA;k_A);K_M)
```

Then M pretends to be A to B...

```
1'. M -> B: Enc(kAM,tA,Sign(kAM,tA;k_A);K_B)
```

And now maybe B discloses secrets to M, or mistakenly trusts information as having come from A, etc.

Naming

Intended meaning: *"At time tA, principal A says that kAB is a good key for communication between A and B"*

Improved protocol:

```
1. A -> B:
     Enc(A,B,kAB,tA,Sign(A,B,kAB,tA;k_A);K_B)
```

Cryptography

Principle: Be clear about what cryptographic primitives are being used, and why, and what properties of them are needed

- Do you need confidentiality?
  - How strong does it need to be?
  - Who should be allowed to learn secrets?
  - What algorithms are acceptable? Are any unacceptable?
- Do you need integrity? (similar questions)
- Do you need both?

Cryptography

"There is considerable confusion about the uses and meaning of encryption" [Abadi & Needham]

- Sometimes (correctly) used for confidentiality
- Sometimes used incorrectly for integrity
  - Sometimes used incorrectly to bind parts of messages, i.e., prevent splicing
  - But Enc(X,Y) might turn out to be exactly the same as Enc(X),Enc(Y), depending on the exact Enc in use
- Confusing notation in literature: $\{m\}_k$
  - Sometimes used to unify notions of Enc(m; k) and MAC/Sign(m; k)
  - Then hard to discern what properties the protocol designer wanted of that primitive

Cryptography

Principle: A principal who signs a message that is already encrypted can't be assumed to know the plaintext of that message

- From Sign("I like ice cream"; k_A), safe to conclude A claims to like ice cream
- From Sign(Enc("I like ice cream"; k); k_A), not safe to conclude that fact, because A might not have access to k

Cryptography

ISO/IEC 11770-3 Key Transport Mechanism 2:

```
1. A -> B: B, tA, Enc(A,kAB; K_B),
   Sign(B, tA, Enc(A,kAB; K_B); k_A)
```

Nothing guarantees that A actually knows the session key kAB

- `Enc(A,kAB; K_B)` could have been given to A by the attacker
- So protocol does not provide key confirmation
- B must trust A not to sign unknown keys

  (or, maybe, trust that if A does so, anyone else who knows the key is at least as trustable as A)

Cryptography

A similar issue:

```
1. A->B: Enc(m; K_B), Sign(m; k_A)
```

which, recall, almost always practically means:

```
1. A->B: Enc(m; K_B), Sign(H(m); k_A)
```

Nothing guarantees that A actually knows m

- Enc(m; K_B) and H(m) could have been given to A by the attacker
- So the protocol does not guarantee plaintext knowledge

Cryptography

Moral of the signing story:

- Be wary if a protocol ever asks a principal to sign something that is already encrypted or hashed
- Be wary if a protocol ever asks a principal to sign something that was received from someone else

Freshness

Principle: Be clear what properties are assumed of nonces

- unique? unpredictable?
- counters can guarantee uniqueness, not unpredictability
- predictable nonces are subject to replay or preplay

Principle: Don't use nonces in place of names

- make principles restate their names for clarity of message, not just present a nonce that supposedly only they would know

Principle: If timestamps are used as nonces, then:

1. The difference between local clocks must be much less than the allowable age of a message
2. The time synchronization mechanism becomes part of the TCB

Principle: A key that has been used recently might be old and compromised

- as Protocol 4, attack 4 in this lecture demonstrates

Trust

Principle: State what trust assumptions are necessary, and why

Examples:

- Server must be trusted to issue correct timestamps
- Principal must be trusted to choose good keys

...applies to all of computer security!

Upcoming events

- [today] make sure you've decrypted A3
- [next Wed] A3 due

Anyone who considers protocol unimportant has never dealt with a cat. – Robert A. Heinlein