




# Breaking and Fixing Public-Key Kerberos

Iliano Cervesato

Carnegie Mellon University - Qatar  
`iliano@cmu.edu`

**Joint work with** Andre Scedrov, Aaron Jaggar, Joe-Kai Tsay, Christopher Walstad

# Outline

- 
- Kerberos 5
    - PKINIT
  - Breaking PKINIT
  - Fixing PKINIT
  - Developments



# The Kerberos Verification Project

- Started in 2001
  - Test MSR on a **real** protocol
    - Kerberos 5 was gaining popularity
- 2002-03: detailed analysis of main protocol
  - Kerberos 5 behaves as expected
    - Authentication and confidentiality properties hold
    - Some anomalous behavior, but not attacks
      - One still under review in the IETF Working Group
- 2004: cross-realm authentication
  - Detailed analysis of what can go wrong if uncheckable hypothesis not met
- 2005: public-key extension of Kerberos - PKINIT
  - **Serious attack**
- Close, ongoing interactions with IETF WG

# Kerberos



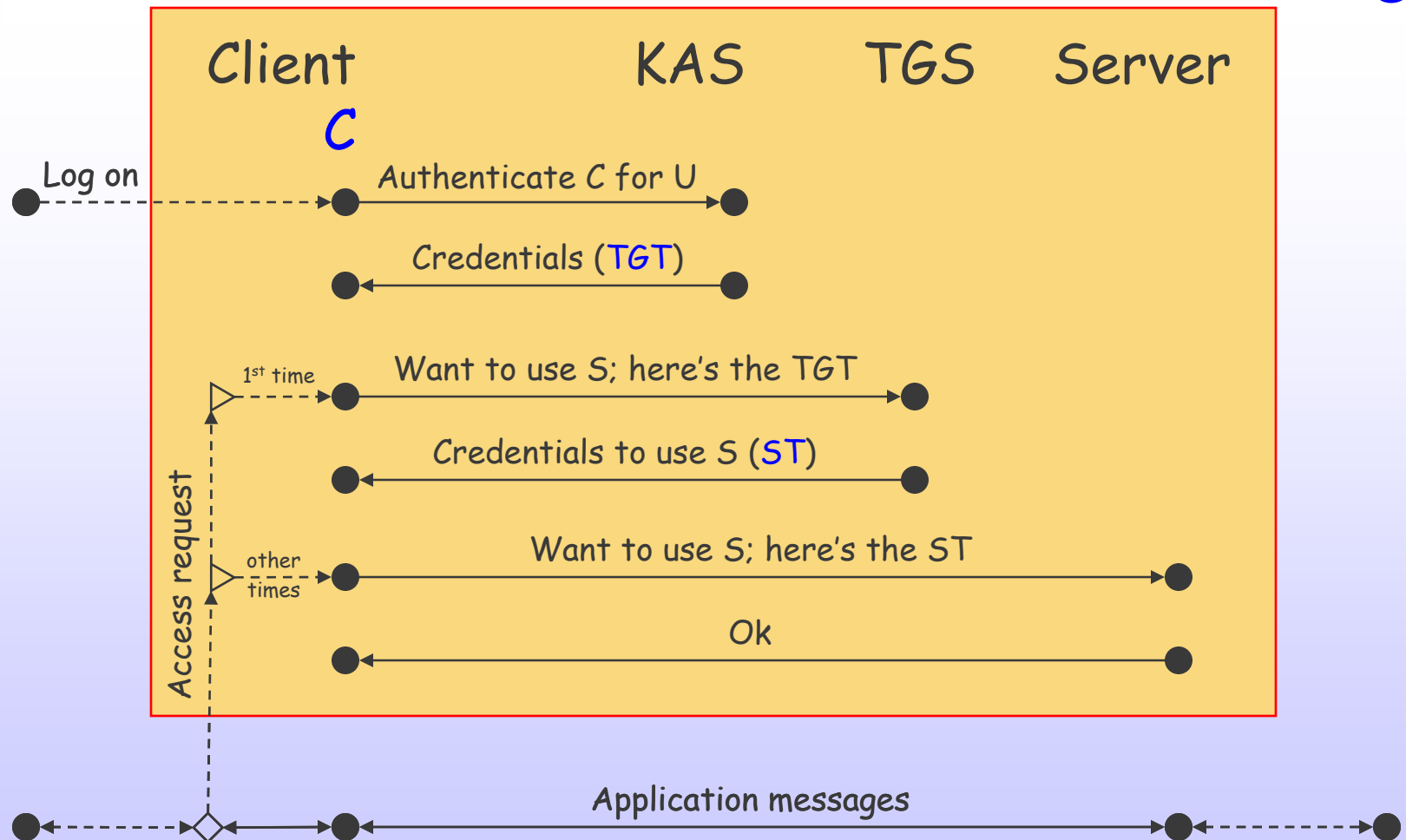
- Goals
  - Repeatedly authenticate a client to multiple servers
    - Remote login, file access, print spooler, email, directory, ...
  - Transparent to user
- History
  - Kerberos 4: 1989 - now (less and less)
  - Kerberos 5: 1993 - now (more and more)
    - Developed by IETF
      - Members from across industry
      - Define interoperability standards
    - 10 active documents, over 350 pages
    - This is a live protocol
      - New extensions under development in IETF WG
- A real world protocol
  - Part of Windows, Linux, Unix, Mac OS, ...
    - Microsoft will phase out all other authentication technology
  - Cable TV boxes, high availability server systems, ...

# Basic Kerberos Operation

User  
U

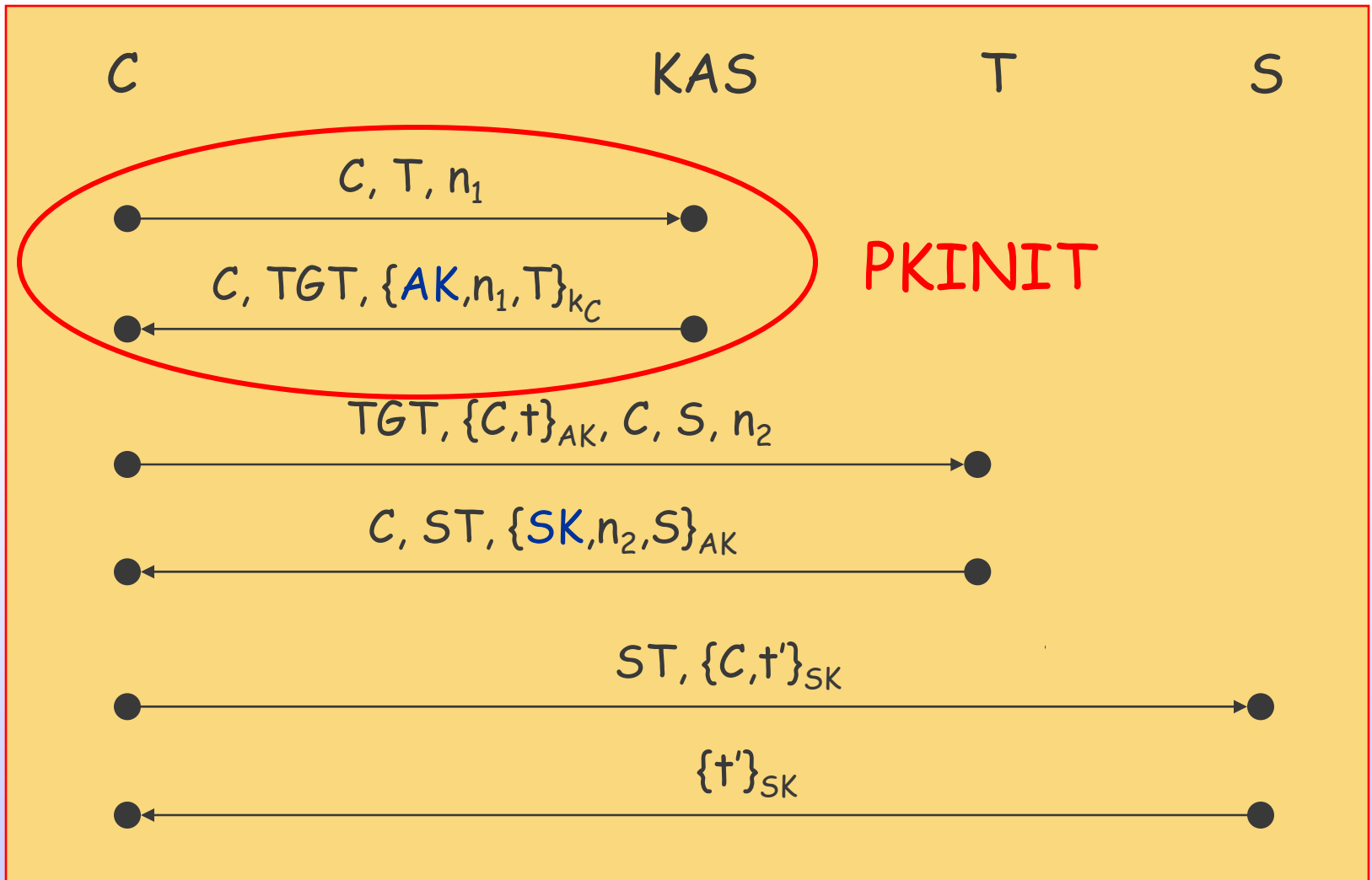
Kerberos

Service  
S



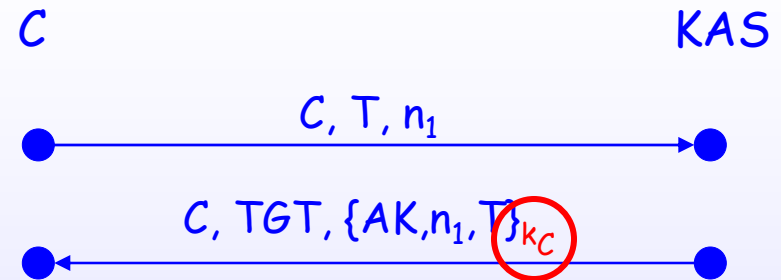
# Abstract Messages

$TGT = \{AK, C\}_{k_T}$   
 $ST = \{SK, C\}_{k_S}$



# Public-Key Kerberos

- Extend basic Kerberos 5 to use Public Keys
  - Change first round to avoid long-term shared keys ( $k_C$ )



- Motivations
  - Security
    - Avoid use of password-derived keys
      - Smartcard authentication support
    - If KAS is compromised, don't need to regenerate shared keys
  - Administrative convenience
    - Avoid the need to register in advance of using Kerberized services
    - Delegate management of keys to external PKI



# PKINIT Revisions

- Now RFC 4556
- Then, a series of IETF Drafts
  - Last, -34
  - We found attack in -25 (May 2005)
    - We analyzed -26
    - Traced back to -00 (1996)
  - Attack fixed in -27 (July 2005)
- Widely deployed
  - All versions of Windows since Win2K
  - Linux since 2003 (Heimdal implementation)
  - Domain specific systems
    - CableLabs implementation for TV cable boxes, ...
  - Under development for MIT reference implementation
    - Unix, Mac OS, ...







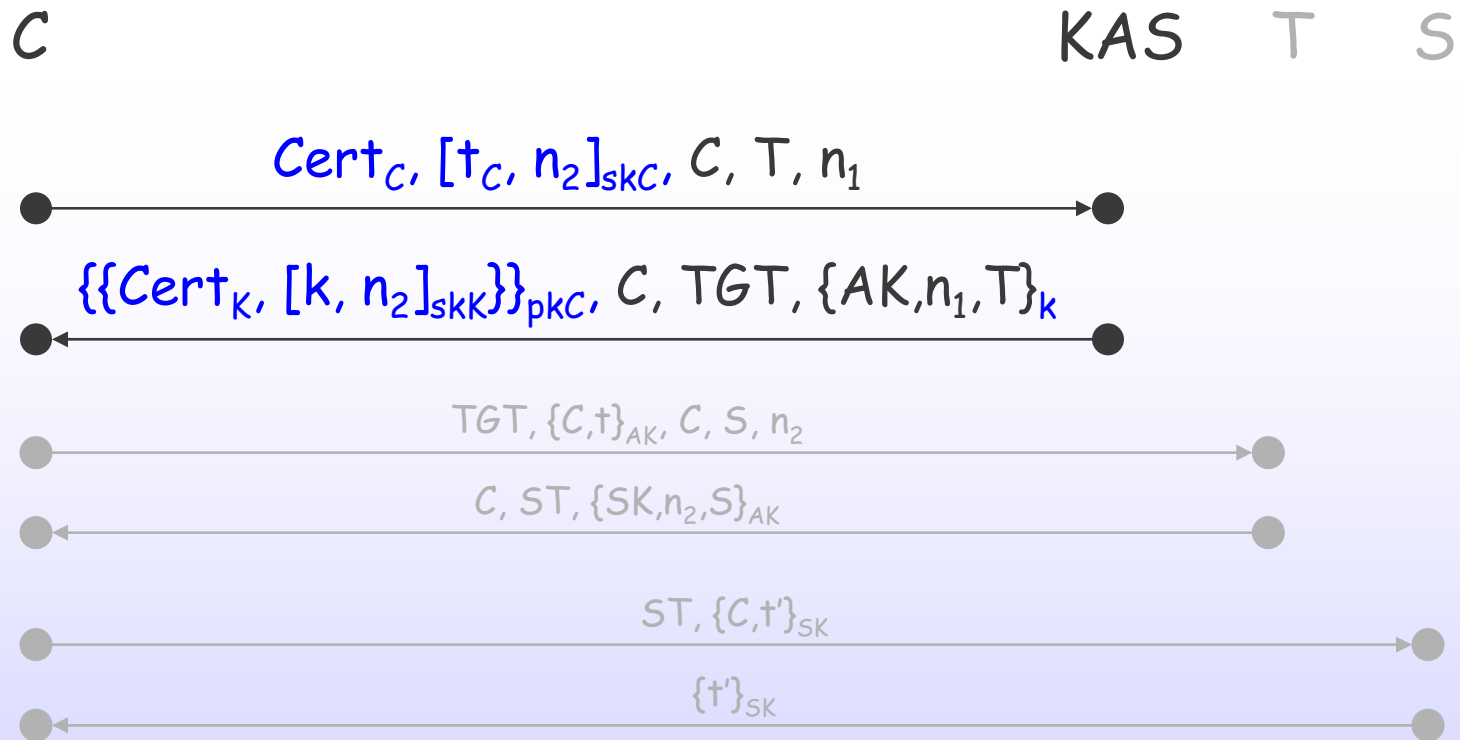
# Two Modes

No more key  $k_c$  shared between C and KAS

- Credentials for C encrypted under a temporary key  $k$ 
  - How to generate and deliver  $k$ ?
- Public-key encryption
  - $k$  is generated by KAS
  - $k$  encrypted under C's public key and signed by KAS
  - Attack is against this mode
- Diffie-Hellman
  - $k$  is derived from DH exchange between C and KAS
  - C and KAS each send signed data contributing to DH key
    - Option for 'reuse' of the shared secret
  - Not widely implemented
    - CableLabs appears to be only implementation of DH mode
  - Initial inspection did not turn up attacks against this mode

# PKINIT in PKE-mode

$$\begin{aligned} \text{TGT} &= \{AK, C\}_{k_T} \\ \text{ST} &= \{SK, C\}_{k_S} \end{aligned}$$



- $\{m\}_k$ : shared-key encryption
- $\{\{m\}\}_{pk}$ : public-key encryption
- $[m]_{sk}$ : digital signature

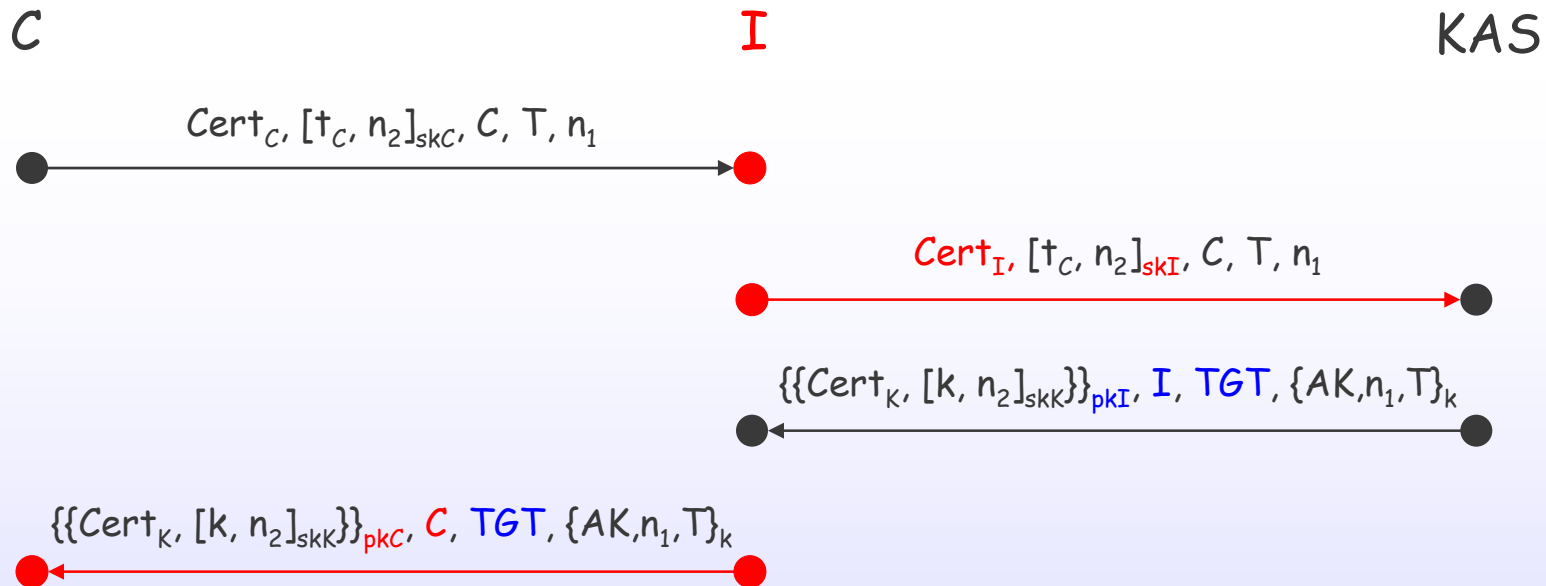
# Outline

- Kerberos 5
  - PKINIT
- **Breaking PKINIT**
- Fixing PKINIT
- Developments



$$TGT = \{AK, I\}_{k_T}$$

# The Attack



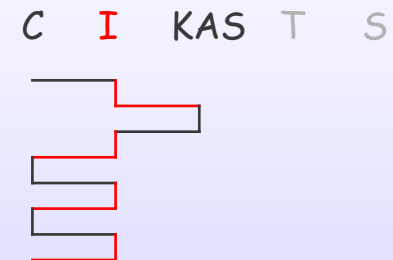
- Failure of authentication
  - C believes to be talking to KAS, is talking to I instead
- Failure of confidentiality
  - I knows AK (and k)
    - C believes KAS produced AK and k just for her

# After the First Round ...

- **I** repeats attack on follow up exchanges
  - Monitors communications
  - Learns keys in replies



- **I** impersonates servers
  - Forge reply messages
  - T, S not involved



- Mixed strategy



# Notes about this Attack

- This is a deterministic attack
  - Conducted at symbolic Dolev-Yao level
  - Man-in-the-middle attack
- **I** must be a legal user
  - Otherwise, KAS would not talk to him
- **C** is authenticated to **S** as **I** (not as **C**)
  - **I** does not trick **S** to believe he is **C**
    - **I** can observe all communications between **C** and **S**
    - **I** can pretend to be **S** to **C**
- DH mode appears to avoid this attack
  - Still need to formally prove security for DH



# Outline

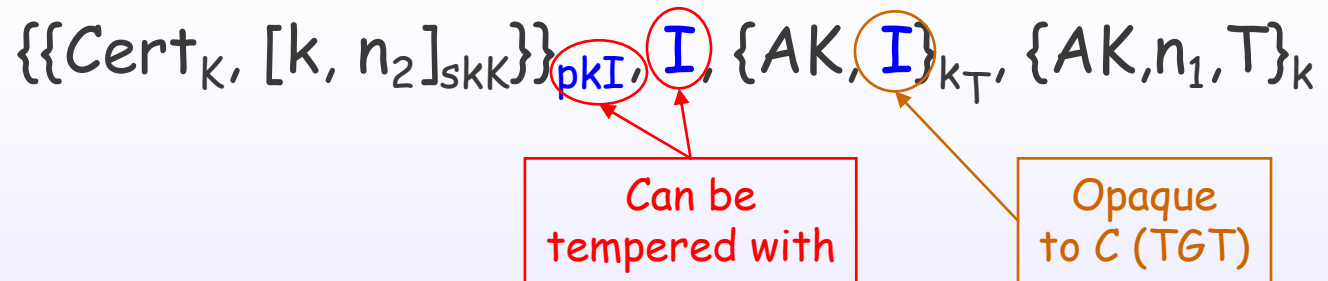
- Kerberos 5
  - PKINIT
- Breaking PKINIT
- Fixing PKINIT
- Developments





# What Went Wrong?

- C cannot tell the reply was not for her



➤ Misbinding of request and reply

- I can
  - Tamper with signature in request
  - Tamper with encryption in reply



# A Familiar Attack ...

- Tampering with signatures
  - 1992: Signature-based variant of StS [Diffie, van Oorschot, Wiener]
  - 2003: basic authenticated DH mode in IKE [Canetti, Krawczyk]
- Tampering with encryption
  - 1996: Needham-Schroeder public key protocol [Lowe]
- Tampering with both
  - 1995: SPLICE/AS [Hwang, Chen] [Clark, Jacob]
- Our attack is the first instance in a widely deployed real-world protocol



# Desired Authentication Property

If a client **C** processes a message containing KAS-generated public-key credentials, then the KAS produced such credentials **for C**

- The attack shows this property does not hold in PKINIT-00/-26
- What are the necessary conditions for the property to hold?

# General Fix

- Sign data identifying client

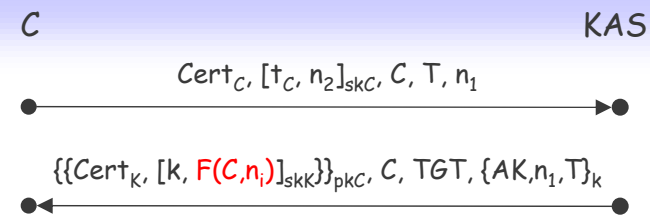
- The KAS signs  $k, F(C, n_i)$ 
  - Either  $n_1$  or  $n_2$  (or both)
- Assume  $F(C, n) = F(C', n')$  implies  $C = C'$  and  $n = n'$

- We have formally proved that this guarantees authentication

- $n_2$  is redundant

- Further questions

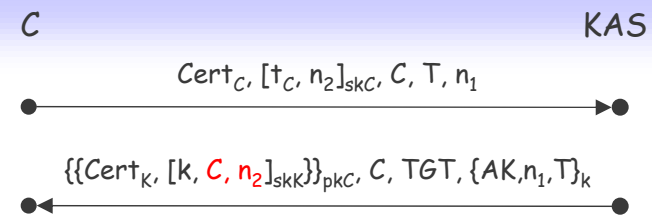
- Does  $cname/crealm$  uniquely identify client?
- Added secrecy properties if  $F(C, n_i)$  identifies  $pkC$ ?



# Initial Proposal

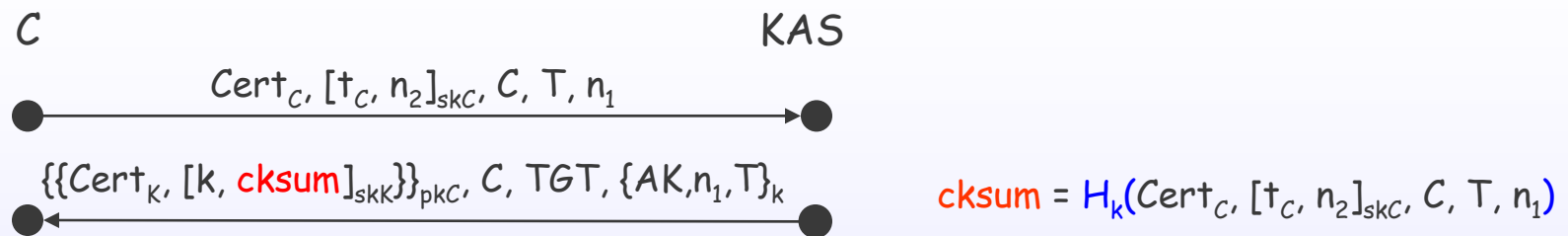
- Traditional approach

$$F(C, n_i) = C, n_2$$



# Fix Adopted by Kerberos WG

$F(C, n_i) = \text{Keyed hash of request}$



- E.g.,  $H = \text{hmac-sha1-96-aes128}$
- Why??
  - Easier to implement than signing  $k, C, n_2$
- Included in PKINIT-27
- Formal assumptions
  - $H$  is preimage resistant
  - KAS's signature key is secret

# Outline

- Kerberos 5
  - PKINIT
- Breaking PKINIT
- Fixing PKINIT
- **Developments**





# Timeline

- **Early May '05:** Top Kerb. WG members notified
  - Request to hold off full disclosure
- **Late May:** fixes proposed
- **June:** Microsoft reproduces attack
  - Hold off any disclosure
- **July:** Kerberos WG notified
- **July:** IETF adopts fix
- **July:** PKINIT-27 incorporates it
- **Aug.:** Attack reported in MS Security Bulletin
- **Oct.:** Patch available for Heimdal (Linux)



# Real-World Impact

- Design vulnerability on widely deployed protocol
- Immediate responses
  - IETF fix to specification
  - Microsoft patch  
<http://www.microsoft.com/technet/security/bulletin/MS05-042.msp>
  - Linux patch
  - CERT entry  
<http://www.kb.cert.org/vuls/id/477341>
- Request to IETF developers to seek formal validation of protocols




# Interactions with IETF

- Close collaboration with IETF Kerberos WG
  - Discussed possible fixes we were considering
  - Attack announced on WG list in July
  - We verified a fix the WG suggested
    - This was incorporated into PKINIT-27
  - Presented this work at IETF-63
    - Discussed possible fixes and our analysis of these
    - Useful discussions with WG participants on other areas for work
  - Now regular participants at IETF / krb-wg meetings
- Impact of formal methods in IETF security area
  - At security-area level, they want to see more interaction with formal methods



# Conclusions

- 
- Extended formalization of Kerberos 5 to PKINIT
  - Serious attack against public-key encryption mode in PKINIT-00/-26
    - Protocol-level attack with real-world effects
    - General fix defending against this
  - Close collaboration with IETF WG
    - Discussion and analysis of possible fixes
      - We've analyzed the fix employed in PKINIT-27

# Future Work

- Fully analyze and verify PKINIT
  - Computational proofs
    - *E.g.*, signature strength
  - Look at DH mode
- Other parts of Kerberos suite
  - Password changing subprotocol
- Continue interactions with WG
- Timed analysis

