Computationally Sound Mechanized Proofs of Basic and Public-key Kerberos

Protocal Exchange, Oct 26th 2007

B. Blanchet[1], A. D. Jaggard[2], A. Scedrov[3], J.-K. Tsay[3]

[1]École Normale Supérieure, [2]Rutgers University, [3]University of Pennsylvania

Context

Analysis of Cryptographic Protocols

Mechanized Proofs

Symbolic/
Dolev-Yao

- Algebra of terms
- Good for checking protocol structure
- Limited adversary capabilities

Using strong Crypto

Computational

- Complexity theory
- Probability theory
- Strong security guarantees

Academic
Protocols

e.g.
- NSL
- Otway-Rees
- Yahalom

Commercial
Protocols

Kerberos PKINIT

e.g.
- TLS
- Kerberos
- IKE

Hand proofs in Computational model prone to human error, and
even in Dolev-Yao model highly time consuming for more complex protocols

Overview (1)

- Formalization and Analysis of Kerberos 5 with and without its public-key extension PKINIT (in Public-Key mode), a public-key extension to Kerberos 5, using the CryptoVerif tool

- First computationally sound mechanized proof of a full industrial-sized protocol
  - Especially PKINIT is complex, involving both asymmetric and symmetric cryptographic primitives
  - Kerberos and PKINIT are available for all major operating systems, e.g. implemented in Microsoft Windows (Vista/XP/2000) and Windows Server 2003

- Generalization of Key Usability notion

Overview (2)

- Part of an ongoing analysis of Kerberos 5 suite
  - Previously discovered a flaw in a draft version of PKINIT used in Windows (XP/2000) and Windows Server 2003
    - Joint work with Cervesato and Walstad
  - Previously conducted by-hand computational proofs of PKINIT and Kerberos
    - Joint work with Cervesato and Backes using the *Backes-Pfitzmann-Waidner model (BPW)*
- CryptoVerif tool works directly in the computational model
  - So far tested only on *academic* protocols, e.g. NSL, Otway-Rees, Yahalom
  - Our work provides evidence for the suitability of CryptoVerif for *industrial* protocols

Related Protocol Work

- [Butler, Cervesato,Jaggard, Scedrov,Walstad '02, '03, '06], [Cervesato,Jaggard,Scedrov,Tsay,Walstad '06]: Symbolic analysis of Kerberos (basic and public-key) using Multi Set Rewriting (Includes the attack on PKINIT draft version)
- [Backes,Cervesato,Jaggard,Scedrov,Tsay '06]: Computational Sound by-hand Proofs of Kerberos using the BPW model
- [He,Sundararajan,Datta,Derek,Mitchell '05]: By-hand symbolic correctness proof of IEEE 802.11i and TLS using Protocol Composition Logic
- [Roy,Datta,Derek,Mitchell '07]: By-hand correctness proofs of Kerberos (incl. Diffie-Hellman mode of PKINIT) using Computational Protocol Composition Logic
- [Meadows '99] : Symbolic analysis of IETF IKE with NRL protocol analyzer
- [Bella,Paulson '97] / [Paulson '97]: Symbolic analysis with Isabelle theorem prover of Kerberos 4 / TLS

…

More Mechanized Prover Background

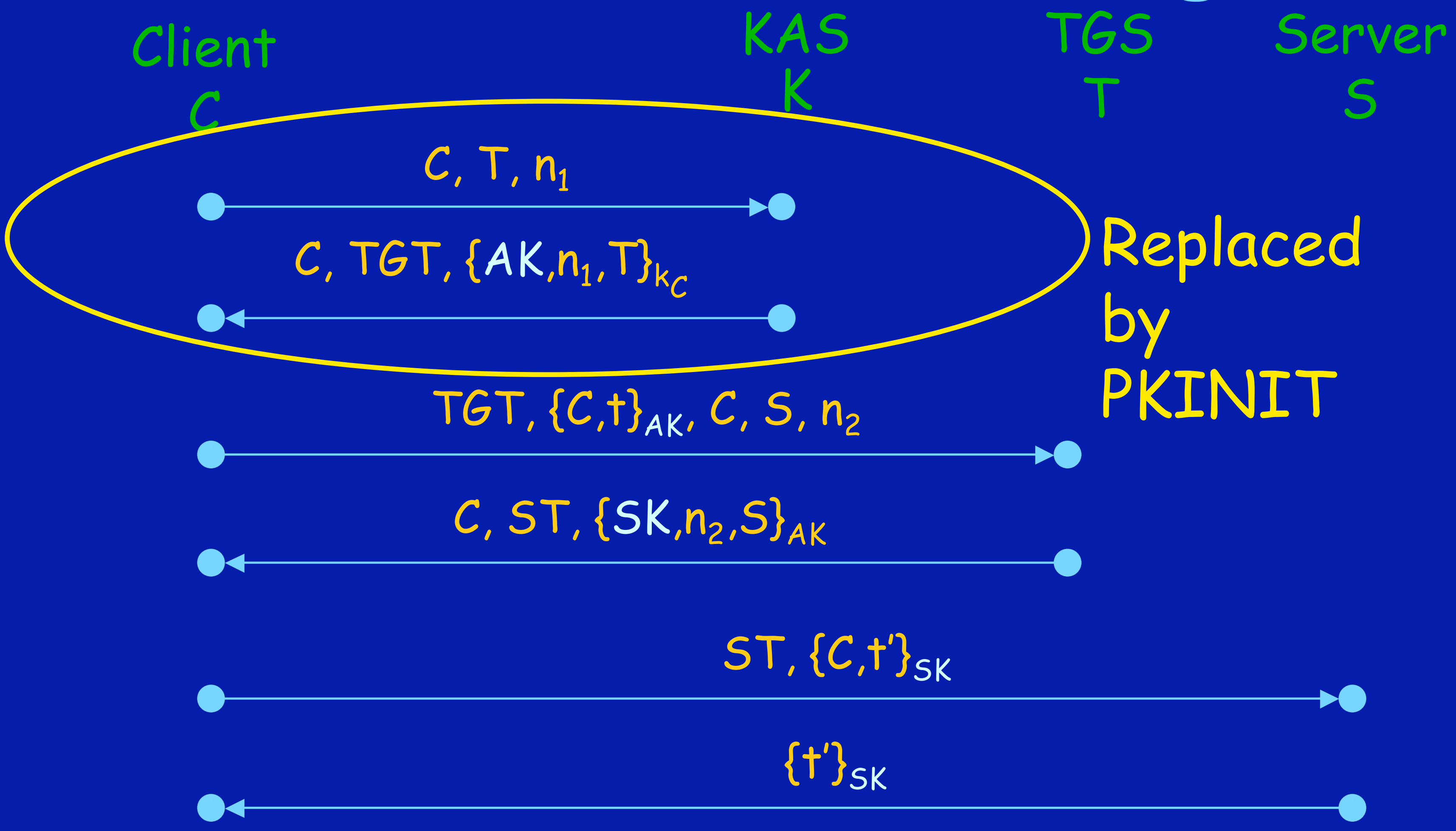
- [Blanchet'06,'07], [Blanchet,Pointcheval '06]: CryptoVerif; computationally sound mechanized prover
- [Backes,Basin,Pfitzmann,Sprenger,Waidner '06]: Beginnings of automation of BPW using Isabelle theorem prover
- [Armando,Basin,Boichut,Chevalier,Compagna,Cuellar,Hankes Drielsma,Heám,Kouchnarenko,Mantovani,Mödersheim, von Oheimb,Rusinowitch,Santiago,Turuani,Viganò,Vigneron '05]: AVISPA tool for automated symbolic validation of protocols and applications
- [Blanchet '04]: ProVerif; automatic Dolev-Yao verification tool
- [Cremers '06]: Scyther; automatic Dolev-Yao verification tool
- [Cortier,Warinschi '05]: Computationally sound, automated symbolic analysis using Casrul tool

…

Kerberos Overview

- Goals
  - Repeatedly authenticate a client to multiple servers on single log-on
    - Remote login, file access, print spooler, email, directory, …
- A real world protocol
  - Part of Windows, Linux, Unix, Mac OS, …
  - Cable TV boxes, high availability server systems, …
  - Standardization and ongoing extension/refinement by IETF (very active --- 10 documents)
  - RFC 4120, RFC 4556, …

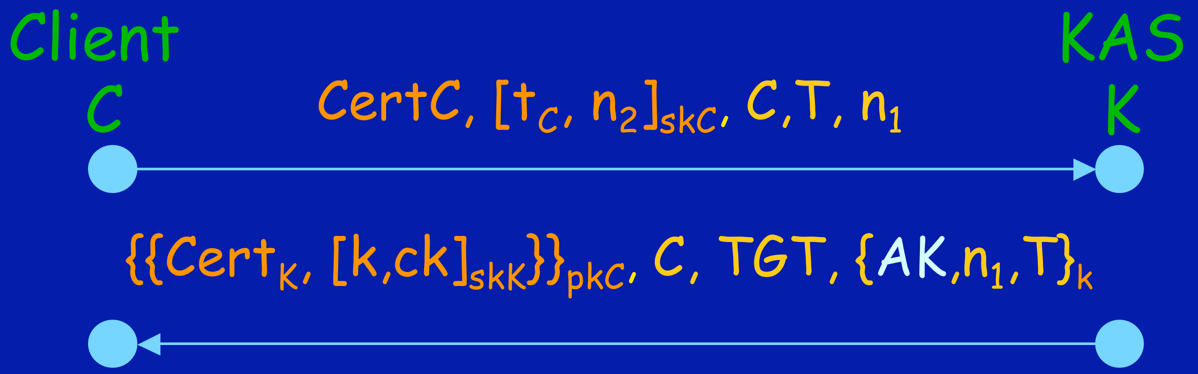
Abstract Kerberos Messages


Client
C
KAS
K
TGS
T
Server
S
C, T, n_1
$C, TGT, \{AK, n_1, T\}_{k_C}$
Replaced by PKINIT
$TGT, \{C, t\}_{AK}, C, S, n_2$
$C, ST, \{SK, n_2, S\}_{AK}$
$ST, \{C, t'\}_{SK}$
$\{t'\}_{SK}$

$TGT = \{AK, C, t_K\}_{k_T}$

$ST = \{SK, C, t_T\}_{k_S}$

Public-Key Kerberos


Client
C
KAS
K
CertC, $[t_C, n_2]_{skC}$, C,T, n_1
$\{\{Cert_K, [k,ck]_{skK}\}\}_{pkC}$, C, TGT, $\{AK, n_1, T\}_k$

$\boxed{TGT = \{AK, C, t_K\}_{k_T}\ ,\ ck = Hash(k, CertC, [t_C, n_2]_{skC}, C, T, n_1)}$

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

- Motivations
  - Administrative convenience: Avoid the need to register in advance of using Kerberized services
  - Security: Avoid use of password-derived keys
    - Smartcard authentication support instead

Cryptographic Assumptions

- Public-key encryption assumed to be IND-CCA2, signature scheme assumed to be UF-CMA
- Symmetric encryption implemented as *encrypt-then-MAC*, with IND-CPA encryption and (W)UF-CMA message authentication code
  - This implies IND-CCA2 and INT-PTXT [Bellare,Namprempre'00]
- Hash function is collision resistant

Authentication (1)

- We can show with CryptoVerif that following holds with overwhelming probability

1. Authentication of the KAS to the client [inj]
    - If an honest client receives what appears to be a valid reply from the KAS, then the KAS generated a reply or the client
2. Authentication of request for ST
    - If an honest TGS processes a valid request for a service ticket ST, then the ticket in the request was generated by the KAS and the authenticator included in the request was generated by the honest client (modulo the MACs).
3. Authentication of TGS to client [inj]
    - If an honest client sees that appears to be a valid reply to a request for a ST for an honest server S from an honest TGS, then the TGS generated a reply for the client.

Authentication (2)

4. Authentication of request to server
    - If an honest server S processes a valid request, ostensibly from an honest client C, containing a service ticket ST and a session key pair (SK, mSK), then some honest TGS generated (SK, mSK) for C to use with S and also created ST (modulo the MAC). Furthermore, C created the authenticator (modulo the MAC).
5. Authentication of server to client
    - If an honest client C sees a valid reply from an honest server S, then this reply was generated by S (modulo the MAC).

Key Secrecy

1. Secrecy AK
   - If an honest client C finishes an AS exchange with the KAS, where the KAS generated the authentication key pair (AK, mAK) for the use between C and an honest TGS T, then AK and mAK are secret w.r.t. the *real-or-random* definition of secrecy
2. Secrecy of SK
   - If an honest client finishes a TG exchange with an honest TGS, where the TGS generated the service key pair (SK, mSK) for the use between C and an honest server S, then SK and mSK are secret with respect to the *real-or-random* definition of secrecy

- Note: The keys AK and SK will no longer be indistinguishable from random once they are used in a client C's request to the TGS T and the server S, respectively

Key Usability

- Notion of *Key Usability* introduced by Datta, Derek, Mitchell, and Warinschi in 2006
- Weaker than key indistinguishability
- Important for protocols that perform operations with a key during a run and allow for the future use of this key
- An exchanged key is *usable* if it is `good' for future cryptographic operations
    - Definition parallels definition of key indistinguishability
    - Two phase attacker ($\mathcal{A}_e$, $\mathcal{A}_c$): first $\mathcal{A}_e$ interacts with protocol sessions, then $\mathcal{A}_c$ tries to win an attack game that uses exchanged key, e.g. IND-CCA2 against an encryption scheme
    - During second phase, $\mathcal{A}_c$ cannot interact with protocol sessions

Key Usability with CryptoVerif

- Stronger version of key usability (w.r.t to IND-CCA2 encryption), where adversary can still interact with uncompleted protocol sessions during the attack game:
  - The adversary $\mathcal{A}$ first interacts with polynomial many protocol sessions
  - At the request of $\mathcal{A}$, a session id *sid* is drawn at random and $\mathcal{A}$ is given access to LR-encryption oracle $\mathcal{E}_k$ and a decryption oracle $\mathcal{D}_k$, where k is the key locally output in *sid*
  - $\mathcal{A}$ plays variant of an IND-CCA2 game where
    - $\mathcal{A}$ may interact with uncompleted protocol sessions
    - But all sessions of the protocol do not accept ciphertexts output by $\mathcal{E}_k$ when they reach a point of the protocol at which at least one session expects to receive a message encrypted under the key k
- Discussion:
  - Stronger notion (at the very least)
  - More realistic ?
  - Yet another definition of key usability (+ Comp Thm) ?

Key Usability in Kerberos

1. Usability of AK
   - If an honest client C finishes a session of basic or public-key Kerberos involving the KAS and an honest TGS, then the authentication key pair (AK, mAK) is (strongly) usable for IND-CCA2 secure encryption (under mentioned crypto assumptions)

2. Usability of SK
   - If an honest client C finishes a session of basic or public-key Kerberos involving the KAS, an honest TGS, and an honest server S, then the session key pair (SK, mSK) is (strongly) usable for IND-CCA2 secure encryption (under mentioned crypto assumptions)

CryptoVerif (1)

- CryptoVerif (CV) can prove secrecy properties and correspondence asssertions for cryptographic protocols, and also cryptographic primitives
  - Secrecy w.r.t. real-or-random definition
  - Authentication through [injective] correspondence assertions [inj:] φ ==> [inj:] ψ
  - Proof of cryptographic primitives in the random oracle model
- CV works directly in the Computational Model
  - Protocols represented as processes in calculus inspired by pi-calculus, the calculi by [Lincoln,Mitchell,Ramanathan,Scedrov,Teague '98, '99, '02] and [Laud '05]; with probabilistic semantics
  - Processes Q and Q' are *observationally equivalent* (Q≈ Q') if, intuitively, an adversary has negligible probability of distinguishing Q from Q'

CryptoVerif (2)

- Proofs as sequences of games
  - Construct sequence $Q_0 \approx Q_1 \approx \ldots \approx Q_{n-1} \approx Q_n$, where Q_0 formalizes the investigated protocol and desired security properties are obvious in Q_n
  - CV uses cryptographic and syntactic transformations to reach Q_j from Q_{j-1}
- Subtleties with crypto assumptions
- Note: CryptoVerif is sound but not complete
  - Properties it cannot prove are not necessarily invalid
  - CV operates in different modes:
    - Automatic mode (if only symmetric crypto is used)
    - Interactive mode (if public-key crypto is used)
      - Requires user to type in commands that determine the next game transformation
- Static corruption of protocol participants

CryptoVerif (3)

Little example:

$Q_C = ! \, ^{i_C <= N} \, c_2[i_C] (h_T : tgs); \text{new } n_1 : nonce;$
$\overline{c_3[i_C]} \langle C, h_T, n_1 \rangle;$
$c_4[i_C] (= C, m_1 : maxmac, mac_1 : macs, m_2 : maxmac, mac_2 : macs);$
$\text{if } check(m_2, mK_C, mac_2) \text{ then}$
$\text{let } injbot(concat1(AK, mAK, = n_1, = h_T)) = dec(m_2, K_C) \text{ in}$
$\text{event } e_C(h_T, n_1, m, m_2) \; ...$

CryptoVerif proves authentication of K to C by proving the query:

$$\text{inj-event}(e_C(T, n, x, y)) \Rightarrow \text{inj-event}(e_K(C, T, n, z, y))$$

- Runtime: Authentication properties of
  - Basic Kerberos: ca. 7 s, 74 game transformations
  - Public-key Kerberos: ca. 1 min 40 s, 124 game transformations

Summary

- Proof of authentication and secrecy properties of basic and public-key Kerberos using the tool CryptoVerif
  - Extended our Kerberos analysis project to include mechanized proofs
- First mechanized proof of authentication and secrecy for a full commercial/real-life protocol directly in the computational model
  - CryptoVerif seems suitable for industrial protocols
- Stronger version of key usability
  - Proved mechanically for Kerberos

Future work

- Using weaker crypto
- Stay closer to Specs
  - Adding additional fields from specs
- Yet another notion of Key Usability ?
- Diffie-Hellman mode of PKINIT
  - Mechanized proof in the computational model
    - Hand Proof exists in Computational PCL [Roy,Datta,Derek,Mitchell '07]

Definition: Strong Key Usability

Let $\Pi=(\mathcal{K}, \mathcal{E}, \mathcal{D}) \in$ S a symmetric encryption scheme, b $\in$ {0,1}, Σ a key exchange protocol, an adversary. Consider following experiment

Exp$^b_{\mathcal{A}, \Sigma, \Pi}(\eta)$:

- First, $\mathcal{A}$ is given η and $\mathcal{A}$ can interact with polynomially many sessions of Σ
- At some point, at the request of $\mathcal{A}$, a session identifier *sid* is drawn at random and $\mathcal{A}$ is given access to a LR-encryption oracle $\mathcal{E}_k$(LR(.,.,b)) and an decryption oracle $\mathcal{D}_k$(.), where k is locally output in *sid.*
- At some point $\mathcal{A}$ plays a variant of an IND-CCA2 attack game
  - Where $\mathcal{A}$ submits same-length pairs to $\mathcal{E}_k$(LR(.,.,b)), never queries $\mathcal{D}_k$(.) on outputs by $\mathcal{E}_k$(LR(.,.,b))
  - A may still interact with uncompleted protocol sessions, but all sessions of the protocol do not accept ciphertexts output by $\mathcal{E}_k$(LR(.,.,b)) when they reach a point in the protocol in which at least one session expects to receive a message encrypted under the key k.
- At some point $\mathcal{A}$ outputs a guess bit d, which is also the output of **Exp**$^b_{\mathcal{A}, \Sigma, \Pi}(\eta)$
- Define the advantage of adversary $\mathcal{A}$ by **ADV**$^{ke}_{\mathcal{A}, \Sigma, \Pi}(\eta)$ = |Pr(**Exp**$^1_{\mathcal{A}, \Sigma, \Pi}(\eta)$ =1) - Pr(**Exp**$^0_{\mathcal{A}, \Sigma, \Pi}(\eta)$ =1)|.
- Then key k is *strongly usable* (for IND-CCA2 encryption) for schemes in S if for all $\Pi \in$ S and all ppt $\mathcal{A}$, **ADV**$^{ke}_{\mathcal{A}, \Sigma, \Pi}(\eta)$ is negligible.