Computationally Sound Mechanized Proof of PKINIT for Kerberos

B. Blanchet\textsuperscript{1}, A. D. Jaggard\textsuperscript{2}, J. Rao\textsuperscript{3}, A. Scedrov\textsuperscript{3}, J.-K. Tsay\textsuperscript{4}

Protocol eXchange Meeting
02 October 2008

\textsuperscript{1}ENS  \textsuperscript{2}Rutgers University  \textsuperscript{3}University of Pennsylvania  \textsuperscript{4}Ruhr-University Bochum
Partially supported by ONR and NSF
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

- 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

• Formalization and Analysis of Kerberos 5 with and without its public-key extension (PKINIT) in “Public-Key mode” using the CryptoVerif tool
• First computationally sound mechanized proof of an industrial-sized protocol
  - PKINIT in particular 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.
• CryptoVerif tool works directly in the computational model
  - Previously tested only on academic protocols, e.g., NSL, Otway-Rees, Yahalom
  - Our work provides evidence for the suitability of CryptoVerif for industrial protocols
Overview

- **Authentication and security results proved using CryptoVerif**
- **Key usability**
  - Define stronger version of IND-CCA2 usability that we can prove for Kerberos using CryptoVerif
  - Now exploring how to define INT-CTXT usability; what’s the right approach?
- **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)
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 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 proof of 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**
Mechanized Prover Background

- [Blanchet'06,'07], [Blanchet, Pointcheval '06]: CryptoVerif; computationally sound mechanized prover
- [Backes, Basin, Pfiztman, 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

• **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 --- 9 current RFCs, 8 drafts)
  - RFC 4120 (Kerberos 5), 4556 (PKINIT), ...
Abstract Kerberos Messages

**Client**

C

**KAS**

K

**TGS**

T

**Server**

S

Authenticate C for U

Want to use S; here's the TGT

Want to use S; here's the ST

Ok

TGT = \{AK, C, t\}_K

ST = \{SK, C, t\}_S

C, T, n_1

C, TGT, \{AK, n_1, T\}_K

TGT, \{C, t\}_AK, C, S, n_2

C, ST, \{SK, n_2, S\}_AK

ST, \{C, t'\}_SK

\{t'\}_SK

Replaced by PKINIT

TGT = \{AK, C, t_K\}_K
ST = \{SK, C, t_T\}_K
Public-Key Kerberos

- **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 shared key to use Kerberized services
  - Security: Avoid use of password-derived keys
  - Smartcard authentication support

\[
\begin{align*}
&\text{Client } C \\
&C \rightarrow KAS \text{ K} \\
&C \qquad \text{Cert}_C, [t_C, n_2]_{sk_C}, C, T, n_1 \\
&\quad \rightarrow \{\{\text{Cert}_K, [k,ck]_{sk_K}\}_{pk_C}, C, TGT, \{AK,n_1,T\} \} \\
&\quad \rightarrow \text{KAS} \text{ K} \\
&TGT = \{AK,C,t_K\}_{k_T}, \text{ ck} = \text{Hash}_k(C, \{t_C, n_2\}_{sk_C}, C, T, n_1) \\
\end{align*}
\]
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 and INT-CTX 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, 2)

- **Authentication in PKINIT**
  - Whenever an honest client $C$ finishes a PKINIT exchange with KAS $K$, after sending a request $m_1$ and receiving a response $\{\{\text{Cert}_K, [k,ck]_{skCC}\}\}_{pkC}, C, \{AK,n_1,T\}_k$ (disregarding the MACS), the KAS $K$ must have received $m_1$ and sent $\{\{\text{Cert}_K, [k,ck]_{skCC}\}\}_{pkC}, C, TGT, \{AK,n_1,T\}_k$ (disregarding the MACS) with overwhelming probability.

  - CryptoVerif proves authentication of $K$ to $C$ by proving the query:
    
    \[
    \text{query } x:\text{bitstring}, y:\text{bitstring}, k:\text{key};
    \]
    
    \[
    \text{event } \text{inj:fullC}(K,k,x,y) \implies \text{inj:fullK}(C,k,x,y).
    \]
    
    where $x=m_1$, $y=\{\{\text{Cert}_K, [k,ck]_{skCC}\}\}_{pkC}, C, \{AK,n_1,T\}_k$ (without macs), and $k=AK$. 

Authentication

- We can show with CryptoVerif that the following hold with overwhelming probability for basic and public-key Kerberos:
  - Authentication (injective) of the KAS to the client
    - If an honest client receives what appears to be a valid reply from the KAS, then the KAS generated a reply for the client.
    - CryptoVerif proves the query:
      \[
      \text{query } x: \text{bitstring}, y: \text{bitstring}, k: \text{key};
      \text{event } \text{inj:fullC}(K,k,x,y) \implies \text{inj:fullK}(C,k,x,y).
      \]
      where \( x = m_1 \), \( y = \{\text{Cert}_K, [k, ck]_{skCC}\}_{pkC}, C, \{AK, n, T\}_k \) (without macs), and \( k = AK \)
  - 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 client (modulo the MACs).
Authentication

• Again, with overwhelming probability for basic and public-key Kerberos:
  - Authentication (injective) of TGS to client
    • If an honest client sees what appears to be a valid reply to a request for a service ticket for an honest server S from an honest TGS, then the TGS generated a reply for the client.
  - Authentication of request to server
    • If an honest server S processes a valid request, ostensibly from an honest client C, that contains 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).
  - 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

• In both basic and public-key Kerberos, we have:
  - Secrecy of AK
    • If an honest client $C$ finishes an AS exchange with the KAS, which generated the authentication key pair $(AK, mAK)$ for use between $C$ and an honest TGS, then $AK$ and $mAK$ are secret w.r.t. the real-or-random definition of secrecy.
  - Secrecy of SK
    • If an honest client finishes a TG exchange with an honest TGS, which generated the service key pair $(SK, mSK)$ for use between $C$ and an honest server $S$, then $SK$ and $mSK$ are secret with respect to the real-or-random definition of secrecy.

• These keys will be distinguishable from random once they are used for encryption in the subsequent requests.
Subsession Key Secrecy

- The final round of Kerberos can be used by C and S to agree on a subsession key for further use.
  - This key can be generated by either the client or the server.
- CryptoVerif proves that both basic and public-key Kerberos preserve:
  - Secrecy of the key possessed by the party that generated the subsession key.
  - One-session secrecy of the key possessed by the other party (of C, S).
  - Difference from possibility of replays:
    - Party accepting (not generating) key might accept same key multiple times, allowing it to be distinguished from random.
    - Current formalization lacks replay cache.
**Key Usability**

- **Notion of key usability** introduced by Datta, Derek, Mitchell, and Warinschi [2006]
- **Weaker than indistinguishability from random**
  - Captures whether a key is still 'good' for future cryptographic operations
- **Important for protocols that perform operations with a key during a run and allow future use of the key**
- **Definition parallels definition of key indistinguishability**
  - Two-phase attacker (Ae, Ac): Ae interacts with protocol sessions, then Ac tries to win an attack game that uses the exchanged keys, e.g., IND-CCA against an encryption scheme
  - During the second phase, Ac cannot interact with protocol sessions
Key Usability with CryptoVerif

- Stronger version of key usability (w.r.t. IND-CCA2 encryption), where adversary can still interact with uncompleted protocol sessions during the attack game:
  - Adversary A first interacts with polynomially many protocol sessions
  - A requests a session id to be drawn at random; let $k$ be the key locally output in that session
  - A is given access to LR-encryption oracle $E_k$ and a decryption oracle $D_k$ corresponding to $k$
  - A plays a variant of the IND-CCA2 game where
    - A may interact with uncompleted protocol sessions
    - But all sessions do not accept ciphertexts output by $E_k$ when they reach a point of the protocol at which at least one session expects to receive a message encrypted under $k$
Key Usability in Kerberos

- **We can use CryptoVerif to prove**
  - **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
  - **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, then the session key pair $(SK, mSK)$ is (strongly) usable for IND-CCA2 secure encryption
(Strong) INT-CTXT Usability

- Not previously defined
- How to define?
  - In standard INT-CTXT game, attacker tries to produce a valid ciphertext that was not generated by the encryption oracle
  - In the game for key usability, the attacker interacts with protocol sessions
    - If these include encryptions by protocol participants, the attacker can trivially win the game
    - This possibility highlighted by use of CryptoVerif
  - Currently check (strong) INT-CTXT usability of a session key $k$ by having the decryption oracle refuse:
    - Ciphertexts produced by the encryption oracle
    - Ciphertexts produced by participants (using any session key)
(Strong) INT-CTX Usability

- CryptoVerif can handle this definition
  - Is it the right one? (Seems reasonable)
- Proofs of INT-CTX usability
  - Proofs of usability of AK and SK
  - These currently require some manual analysis
    - CryptoVerif produces games that include branches in which the attacker wins
    - Show that these branches can never be taken
Weakening Crypto

• Leak content of authenticators
  - \(\{C,t\}_{AK}, C, t\) instead of just \(\{C,t\}_{AK}\)
  - \(\{C,t'\}_{SK}, C, t'\) instead of just \(\{C,t'\}_{SK}\)
  - Suggested by examining by-hand proofs in Dolev-Yao model

• The authentication results still hold for both basic and public-key Kerberos

• The secrecy of the subsession key also still holds for both basic and public-key Kerberos

• Advantage of CryptoVerif---very fine control over crypto assumptions
CryptoVerif (1)

- Developed by Blanchet
- CryptoVerif (CV) can prove secrecy properties and correspondence assertions for cryptographic protocols, and also cryptographic primitives
  - Secrecy w.r.t. real-or-random definition
  - Authentication through [injective] correspondence assertions
    \[ \text{inj:}\varphi \implies \text{inj:}\psi \]
  - 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 \((Q \simeq Q’)\) if, intuitively, an adversary has negligible probability of distinguishing Q from Q’
CryptoVerif (2)

- **Proofs as sequences of games**
  - Construct sequence $Q_0, Q_1, \ldots, Q_{n-1}, Q_n$, 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 $Q_{j-1}$ and such that the new game is negligibly different from the old one.

- **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.
Summary

• Proof of authentication and secrecy properties of PKINIT using the tool CryptoVerif
  - Extended our Kerberos analysis project to include mechanized proofs

• First mechanized proof of authentication and secrecy for a commercial/real-life protocol directly in the computational model
  - CryptoVerif seems suitable for industrial protocols

• Stronger notions of usability
  - Working on right definition of INT-CTX usability
Future work

- Finalize INT-CTXT key usability definitions and proofs
- Using weaker crypto
- Add more details from Kerberos RFCs
  - Adding lots of detail to key generation may cause games to explode
  - Diffie-Hellman mode of PKINIT
    - Mechanized proof in the computational model
      - Hand Proof exists in Computational PCL [Roy,Datta,Derek,Mitchell '07]
- Look for general results about usability and composability
Analysis of Cryptographic Protocols

**Hand Proofs**

Symbolic / Dolev-Yao
- Algebra of terms
- Good for checking protocol structure
- Limited adversary capabilities

Using strong Crypto

Computational
- Complexity Theory
- Probability Theory

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

Commercial Protocols
- e.g.
  - TLS
  - Kerberos
  - IKE
  - 802.11i

Hand proofs in Computational model prone to human error, and even in Dolev-Yao model highly time consuming for more complex protocols
Context (2)

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

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

Commercial Protocols
- PKINIT
- e.g.
  - TLS
  - Kerberos
  - IKE
  - 802.11i