

On Key Assignment for Hierarchical Access Control

Jason Crampton · Keith Martin · Peter Wild

Information Security Group · Royal Holloway · University of London

19th Computer Security Foundations Workshop

Introduction

What is hierarchical access control?

We assume the existence of a set of users U and a set of objects O , a partially ordered set (X, \leq) , and a function $\lambda : U \cup O \rightarrow X$

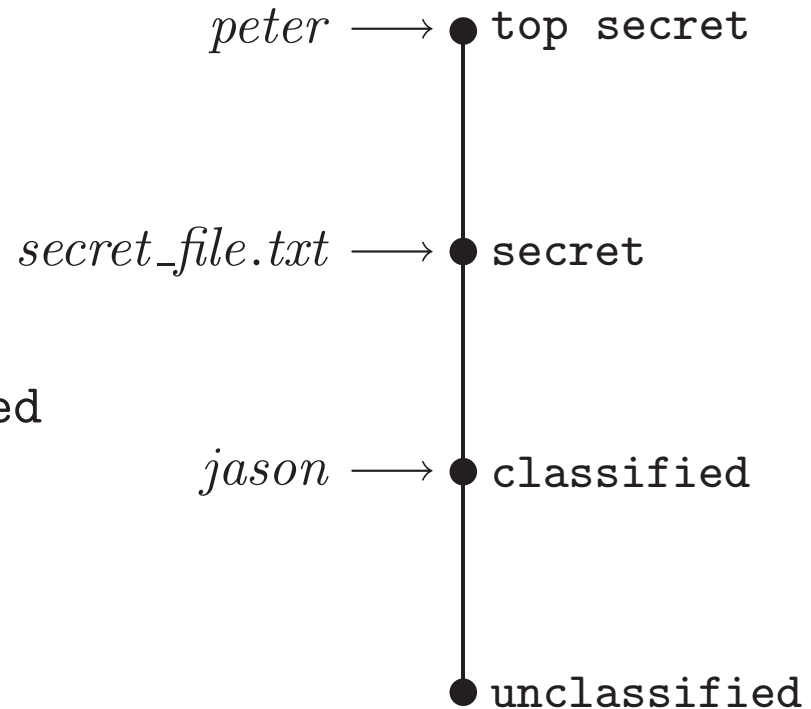
- λ associates each entity e with a security label $\lambda(e)$
- $u \in U$ may access $o \in O$ if $\lambda(u) \geq \lambda(o)$
 - Sometimes known as the *simple security property*
 - Cornerstone of many military security policies

Example

$X = \{\text{unclassified}, \text{classified}, \text{secret}, \text{top secret}\}$

$\text{unclassified} < \text{classified} < \text{secret} < \text{top secret}$

- $\lambda(\text{peter}) = \text{top secret}$,
 $\lambda(\text{jason}) = \text{classified}$
- *peter* can read any object
(including *secret_file.txt*)
- *jason* can read any unclassified
or classified object (but not
secret_file.txt)



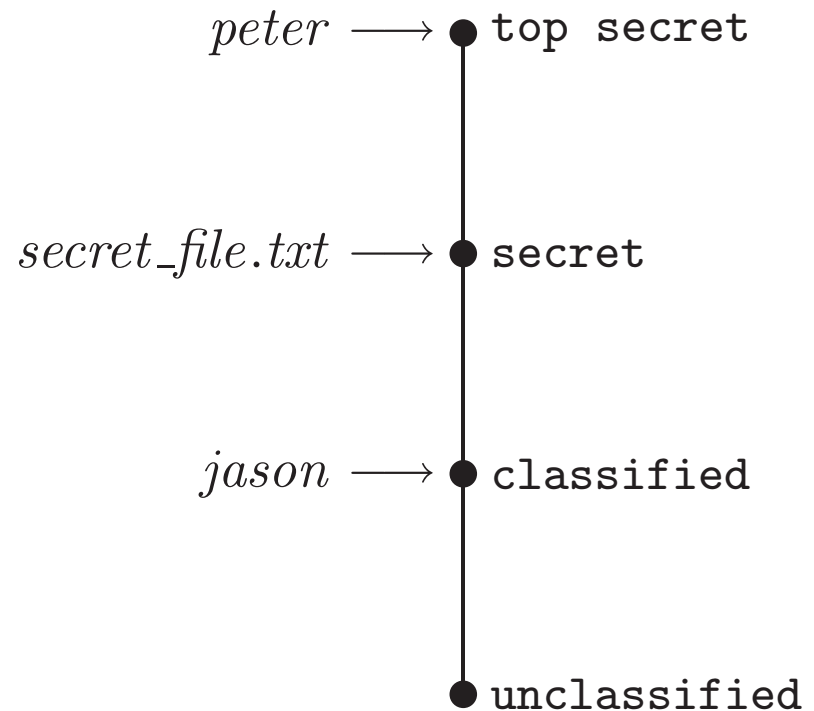
What is a key assignment scheme?

Encrypt objects and supply users with appropriate keys

- Give *peter* k_u , k_c , k_s and k_t
- Give *jason* k_u and k_c

Users have to maintain a number of different keys

- Can we do better?



A simple scheme

Use some form of top-down encryption to generate keys from a security label and the key associated with the parent label

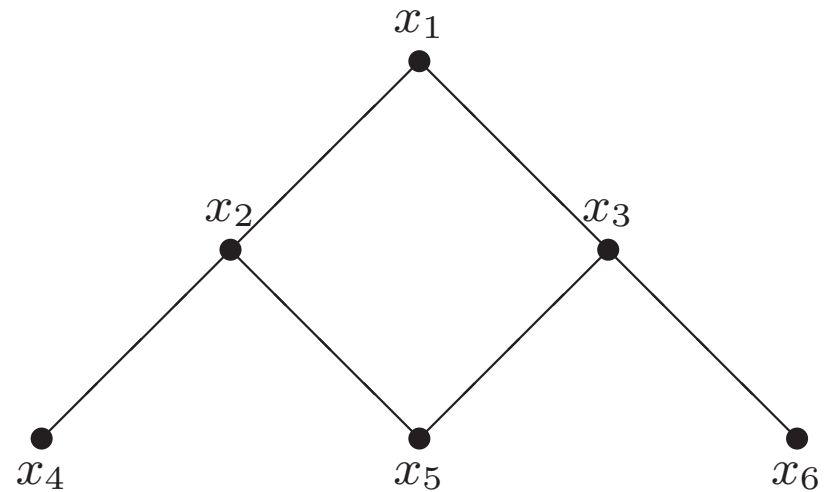
- Choose k_t and define
 - $k_s = E_{k_t}(\text{“secret”})$
 - $k_c = E_{k_s}(\text{“classified”})$
 - $k_u = E_{k_c}(\text{“unclassified”})$
- Give *peter* k_t and *jason* k_c

One implementation is to hash concatenation of parent key and junior security label

Can be extended to a key assignment scheme for trees

General problem

- How do we handle arbitrary posets?
- There is not a unique path from x_1 to x_5



Our motivation

There are (too) many schemes in the literature

- Rely on specific cryptographic primitives
- Do not consider basic requirements and features of key assignment schemes

We want to develop an abstract approach to key assignment schemes

- Classify existing schemes
- Evaluate the respective merits of different types of scheme

Key assignment schemes

Basic concepts

We assume the existence of a scheme administrator (trusted centre)

A key assignment scheme comprises (up to) four algorithms

- `makeKeys` returns a labelled set of encryption keys $(\kappa(x) : x \in X)$
- `makeSecrets` returns a labelled set of secret values $(\sigma(x) : x \in X)$
- `makePublicData` returns a set of data Pub that is made public by the trusted centre
- `getKey` takes $x, y \in X$, $\sigma(x)$ and Pub and returns $\kappa(y)$ whenever $y \leq x$

A scheme has *independent keys* if the keys can be chosen independently of each other and of Pub

Evaluation criteria

- Amount of secret data that needs to be distributed to and stored by end users
- Amount of data that needs to be made public
- Complexity of key derivation
- Complexity of key update (if user leaves or key is compromised)
 - How much secret data needs to be re-distributed?
 - How much public data needs to be re-computed?
- Resistance to collusion attacks

Trivial key assignment scheme

- Independent keys $\kappa(X)$
 - $\sigma(x) = (\kappa(y) : y \leq x)$
 - $Pub = \emptyset$
 - $\kappa(y) \in \sigma(x)$ so key derivation is trivial
- ✗ High private storage costs
 - ✓ No public storage
 - ✗ High update costs for private data
 - ✓ Direct key derivation

Trivial key encrypting key assignment scheme

- Independent keys $\kappa(X)$ and set of key encrypting keys $K(X)$
 - ✗ High private storage costs
- $\sigma(x) = (K(y) : y \leq x)$
 - ✗ High public storage costs
- $Pub = (E_{K(x)}(\kappa(x)) : x \in X)$
 - ✓ Very low costs for update of $\kappa(y)$
- $\kappa(y)$ is obtained by decrypting $E_{K(y)}(\kappa(y)) \in Pub$ using $K(y) \in \sigma(x)$
 - ✗ High costs for update of $K(y)$
 - ✓ Direct key derivation

Direct key encrypting key assignment scheme

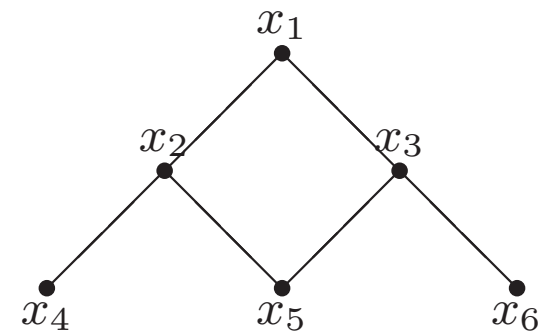
- Independent keys $\kappa(X)$
- $\sigma(x) = \kappa(x)$
- $Pub = (E_{\kappa(x)}(\kappa(y)) : y < x)$
- $\kappa(y)$ is obtained by decrypting $E_{\kappa(x)}(\kappa(y)) \in Pub$ using $\kappa(x)$
- ✓ Minimizes private storage costs
- ✗ High public storage costs
- Moderate costs for update of private and public data
- ✓ Direct key derivation

Iterative key encrypting key assignment scheme

- Independent keys $\kappa(X)$
- $\sigma(x) = \kappa(x)$
- $Pub = (E_{\kappa(x)}(\kappa(y)) : y \triangleleft x)$
- $\kappa(y)$ is obtained by decrypting $\kappa(z)$ for all z on a path from x to y
- ✓ Minimizes private storage costs
- ✓ Minimizes public storage costs
- Moderate costs for update of private and public data
- ✗ Iterative key derivation

Example

- TKAS
 - $\sigma(x_1) = \{\kappa_1, \dots, \kappa_6\}$
- TKEKAS
 - $\sigma(x_1) = \{K_1, \dots, K_1\}$
 - $Pub = \{E_{K_1}(\kappa_1), \dots, E_{K_6}(\kappa_6)\}$
- DKEKAS
 - $Pub = \{E_{\kappa_1}(\kappa_2), E_{\kappa_1}(\kappa_3), E_{\kappa_1}(\kappa_4), \dots\}$
- IKEKAS
 - $Pub = \{E_{\kappa_1}(\kappa_2), E_{\kappa_1}(\kappa_3), E_{\kappa_2}(\kappa_4), \dots\}$



IKEKAS example

Atallah, Frikken and Blanton (*CCS* 2005)

- $Pub = \{\kappa(y) - h(\kappa(x), y) : y \triangleleft x\}$, h is a hash function
- User with security label x can recover $\kappa(y)$ by computing $h(\kappa(x), y)$

Node-based key assignment scheme

- $Pub \supseteq (e(x) : x \in X)$
- $\kappa(x) = f(e(x))$
 - f is a secret function
 - There exists a public algorithm g such that

$$g(f(e(x)), e(x), e(y)) = g(\kappa(x), e(x), e(y)) = \kappa(y)$$

is feasible to compute if and only $y \leq x$

- By construction $\kappa(y)$ can be derived (directly) from $\kappa(x)$ (using g)
- Dependent keys ($\kappa(x) = f(e(x))$)

Example

Akl and Taylor (*ACM Trans. Comp. Sys.*, 1983)

- $Pub = \{n\} \cup (e(x) : x \in X)$
 - $n = pq$, p and q are large primes
 - $e : X \rightarrow \mathbb{N}$ such that $e(x) \mid e(y)$ if and only if $y \leq x$
- $\kappa(x) = s^{e(x)} \pmod n$, where $s \in \mathbb{Z}_n^*$ is a system secret
 - Note that $(s^{e(x)})^{\frac{e(y)}{e(x)}} = s^{e(y)}$
 - Hence $\kappa(y) = (\kappa(x))^{\frac{e(y)}{e(x)}}$
 - It is only feasible to compute $\kappa(y)$ if $y \leq x$ (on the assumption that it is difficult to compute integral roots modulo n)
- Usual to choose $e(x) = \prod_{y \not\leq x} p(x)$, where $p(x)$ is a prime

Characteristics of (simplified) Akl-Taylor scheme

- ✓ Low private storage
 - Moderate public storage
- ✓ Update of public information is very simple
- ✗ Update of secret information worse than IKEKAS
- ✓ Direct key derivation
 - ✗ Exponentiation required

Conclusion

Crude summary

Scheme	Storage		Update $\kappa(x)$		Derivation
	Private	Public	Private	Public	
TKAS	X	✓✓✓	X	✓	✓✓
TKEKAS	X	✓✓	X	XX	✓
DKEKAS	✓	X	✓	XX	✓
IKEKAS	✓	✓	✓	X	X
NBKAS	✓	✓✓	?	✓?	✓?

Schemes in the literature

We surveyed about 30 papers

- 2 are TKAS
- 3 are TKEKAS
- 2 are DKEKAS
- 7 are IKEKAS
- 12 are NBKAS
- A couple of weird hybrids

Often clumsy and almost always over-complicated

Wide variety of cryptographic and mathematical techniques

- RSA
- Rabin cryptosystem
- Polynomial interpolation
- Chinese remainder theorem
- Discrete logs
- Sibling intractable function families
- Hash functions with collisions

Contributions

- Classification of key assignment schemes
 - Provides framework with which to evaluate existing and new schemes
- Improvement to implementation of Akl-Taylor
 - Reduction in key derivation complexity
 - Reduction in storage requirements
 - Improved insight into key updates
- Development of hybrid key assignment scheme
 - Poset “partitioned” into domains
 - Each domain uses a NBKAS
 - Domains treated as “supernodes” in information flow policy and stitched together using an IKEKAS

Future work

- Are there more efficient trapdoor functions for node-based schemes?
- Are there better embeddings of X for Akl-Taylor schemes?
 - Is there a “canonical” representation and embedding for the Bell-LaPadula security lattice?
- Can we extend the model to include keys that only have a limited lifetime?
 - Will need to incorporate some notion of forward secrecy

Questions?